8-2014

Privacy Sensitive Resource Access Monitoring For Android Systems

Leah Xinya Zhao

Privacy Sensitive Resource Access Monitoring For Android Systems

by

Leah Xinya Zhao

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Science
in Computer Engineering

Supervised by

Associate Professor Shanchieh Jay Yang
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
August 2014

Approved by:

Shanchieh Jay Yang, Associate Professor
Thesis Advisor, Department of Computer Engineering

Roy Melton, Senior Lecturer
Thesis Secondary Advisor, Department of Computer Engineering

Wei Le, Assistant Professor
Committee Member, Department of Computing and Information Sciences

Dedication

To my family, my friends, and my professors, who have been there along throughout this whole process.

Acknowledgments

I am grateful for all the time and guidance that Dr. Yang has provided to help me thoughout this thesis process. I would also like to thank Dr. Melton and Dr. Le for the dedication and effort as my committee members.

Abstract

Privacy Sensitive Resource Access Monitoring For Android Systems

Leah Xinya Zhao

Supervising Professor: Shanchieh Jay Yang

Mobile devices, with an extensive array of capabilities and flexibility, are sometimes said to be an extension of the human body. Enhancing device capabilities and incorporating them into everyday life have always been a huge focus of the mobile industry. In the area of mobile data collection, existing works collect various types of user behavior data via mobile device usage, and use the data to aid in further understanding of human behavior. Typical data collection utilizes application or background service installed on the mobile device with user permission to collect data such as accelerometer, call logs, location, wifi transmission, etc. In this process, sensitive user information is tracked through a data tainting process. Contrary to the existing works, this research aims at collecting application behavior instead of user behavior. The goal is to provide a means to analyze how background services access mobile resources, and potentially identify suspicious applications that access sensitive user information.

This investigation proposes an approach to track the access of mobile resources in a real time and sequential way. Specifically, the approach integrates the concept of taint tracking. Each identified user privacy sensitive resource is tagged and marked for tracking. The approach is composed of three different components: collection mechanism, collection client, and collection server. The collection mechanism resides in the Android OS to detect any incoming activity to privacy sensitive mobile resources. Whenever detection occurs, the collection client processes the formatted information. The collection client then communicates with an external server to store the gathered data. From these data, responsible applications, affected

resources, and transmitted data were identified along with sequences of activity resulting from specific user actions. The result is a dynamic, real-time resource for monitoring the process flow of applications. Statistical analysis of sample data collected will be presented to demonstrate some interesting application behaviors and the potential usage of the application behavior data collection process.

Contents

Dedication . . . . . **iii**

Acknowledgments . . . . . **iv**

Abstract . . . . . **v**

1 Introduction **1**
1.1 Motivation and Background 1
1.2 Problem Statement 2

2 Related Work **4**
2.1 Analysis of Mobile Data 4
2.2 Android Architecture 7
2.2.1 Android Application Sandbox 9
2.3 Android Application Analysis 9
2.4 Data Collection Mechanisms 11

3 Methodology **14**
3.1 Overall System Design Analysis 15
3.1.1 Collection Module Analysis 15
3.1.2 Server Module Analysis 17
3.2 Data Collection Components 17
3.2.1 Data Collection Module 20
3.2.2 Collection Client Module 27

3.2.3 Collection Server Module 29
3.3 Test Application Design and Implementation 31

4 Results and Discussions 38
4.1 Experiment Setup 38
4.1.1 Passive Data Collection 39
4.1.2 Active Data Collection 40
4.2 Results 43
4.2.1 Privacy Sensitive Resource Utilization 43
4.2.2 Privacy Sensitive Resource Access Intensity 49
4.2.3 Network Traffic Analysis 55
4.2.4 Action Based Sequential Analysis 61
4.3 Performance 64

5 Conclusion and Future Work 65

Bibliography 68

List of Tables

3.1 Location listeners 22
3.2 Contacts listeners 23
3.3 Microphone listeners 23
3.4 Device information listeners 24
3.5 Camera listeners 25
3.6 Accelerometer listeners 25
3.7 SMS listeners 26
3.8 Browser history listeners 26
3.9 Location based resourse test 32
3.10 Contacts, browser resource, sms based resourse test 33
3.11 Microphone based resource test 33
3.12 Accelerometer based resource test 34
3.13 Camera based resource test 35
3.14 IMEI, IMSI, Phone number, and ICCID based resource test . 36

4.1 Categories and applications downloaded. 39
4.2 User interaction activity time breakdown. 42
4.3 Number of IP address for each application in the passive and active case. 60
4.4 Applications with same IP addresses. 60
4.5 Performance measures of Panorama. 64

List of Figures

2.1 Interaction of students in the MIT community (figure extracted from Dong *et al*, [8]). 5
2.2 Android software stack (figure extracted from Android security overview, [1]). 8
2.3 AppWindow architecture structure (figure extracted from Zhang *at el*,[11]). 10
2.4 NetSense mobile data aquisition infrastructure (figure extracted from Striegel *at el*,[6]). 12
2.5 mFingerprint mobile data collection to application framework (figure extracted from Zhang *at el*,[14]). 13

3.1 Android security structure for user sensitive data access (figure extracted from [1]). 15
3.2 Overall privacy sensitive resource monitoring system design. 18
3.3 Overall data collection module design. 21
3.4 Overall data collection client module design. 27
3.5 Example TaintDroidNotify notification. 28
3.6 Overview of collection server module design and process flow. 29
3.7 MySQL database schema. 30
3.8 SensorTrigger application display. 31

4.1 Number of sensitive resource accesses made by each third-party application passively. 44
4.2 Number of sensitive resource accesses made by each third-party application actively. 45
4.3 Number of sensitive resource accesses made by different categories of third-party application passively. 46
4.4 Number of sensitive resource accesses made by different categories of third-party application actively. 47
4.5 Passive utilization of types of resources in terms of categories of third-party application processes. 48
4.6 Passive utilization of types of resources in terms of categories of third-party application processes. 48
4.7 Intensity heat map of passive resource access attempts. . . . 50
4.8 Intensity heat map of active resource access attempts. . . . 50
4.9 Intensity summary graph of passive resource access attempts. 52
4.10 Intensity summary graph of active resource access attempts. 52
4.11 Intensity of passive resource access attempts by application categories. 53
4.12 Intensity of active resource access attempts by application categories. 53
4.13 Intensity comparison between specific passive and active application categories. 54
4.14 Passive application network traffic. 55
4.15 Active application network traffic. 56
4.16 Passive application network traffic in terms of application categories 57

4.17 Active application network traffic in terms of application categories. 57
4.18 Passive application network traffic in terms of utilized resources. 58
4.19 Active application network traffic in terms of utilized resources. 59
4.20 Passive action sequence of Google location process. 61
4.21 Active action sequence of Google location process. 62
4.22 Action sequence of using Snapchat. 63
4.23 Action sequence of the Weather Channel application. 63

Chapter 1

Introduction

1.1 Motivation and Background

The smartphone industry is a fast growing industry. In 2013 alone, 1.004 billion smartphones were shipped, which is a 38.4 percent increase from 2012 [17]. Accordering to Gartner, a fact-based consulting service, 102 billion applications were downloaded globally in 2013 [24]. With this industry growth, mobile datasets attract interest in many different fields: human behavior analysis, business opportunity exploration, security monitoring, healthcare, and many others. Especially with the fast advancing technology, mobile devices are now capable of performing tasks of various degrees of difficulty. Scenarios exist where applications embed user privacy information collection mechanisms to better learn about all of its users around the world. For researches, mobile data collection mechanism have also been put in place to collect specific datasets for further analysis. The research-based collections are usually explicit. Currently there are many different types of datasets collected and analyzed in which mobile resources such as contacts and sensors were used [20] [10] [28] [27] [26] [21] [18]. However, there are few, if any, datasets collected to understand the dynamic behavior of application processes and their interaction with privacy sensitive mobile resources.

Privacy has always been a hot topic. However, in terms of technology privacy, the rules and boundaries are not very defined. Particularly, on an Android device, before installing the package, the application asks for user permission to gain access to many of the phone resources. This is the only

contract between the user and the application. Once the required permissions have been granted, the application is able to perform tasks without the user's consent regarding to the granted permissions ever again. Many developers utilize this grey area regarding consent to gain understanding of their target market behavior or even more. The Weather app, for example, has a background service to actively utilize the GPS resource to track the user location. Even when the Weather application is not in the foreground, this GPS service will still log the user's location. There is a need to understand these background services behavior to help users understand risks involved with each application.

Another reason to collect application process interaction with privacy sensitive mobile resources is to understand the intensity and frequency of resources accesses at any given period of time. Machine learning algorithms could be applied to the dataset to learn the behavior of a specific phone and observe anomalies in real time. In some other cases, this dataset could be combined with mobile performance datasets to help phone developers better understand each available mobile resources. The sequence of behavior can also be analyzed in real time to identify security breaches. This is a small step towards the cross platform big data challenge.

1.2 Problem Statement

Given the need and benefits of background services accessing mobile resources, there are both security and privacy risks exposed. To address these risks, datasets should be collected relating to activities of background services accessing privacy sensitive mobile resources.

This thesis work proposes a dynamic real-time privacy sensitive mobile resources monitoring system. The idea of data tainting is explored and incorporated into the overall system. The concept is that a flag is assigned to each defined input, and then that flag propagates to all data derived from that input. Another concept that is incorporated along with the tainting process is the idea of active listeners. Active listeners are implemented in the Android framework layer to monitor specific actions performed for specific privacy sensitive resources.

The scope of this work is sectioned into three major components: monitoring mechanism, mobile client, and server storage. Each one of these components will be further discussed in later sections. As with any research work, there are some limitations. For one, the Android ROM must be modifiable to support the system wide collection mechanism implementation. Second, with the stock Android ROM, there is a limitation on the kind of third party applications availability.

The specific details of this thesis investigation are addressed throughout the later chapters. Chapter Two addresses existing works in the fields related to mobile data collection and analysis. Chapter Three is the methodology section, which addresses the design requirements as well as detailed implementation of each component. The results and analysis of this thesis investigation are presented in Chapter Four. Finally, an overall summary and future work are addressed in Chapter Five.

Chapter 2

Related Work

The work involved in this thesis research is motivated by several related areas pertaining to mobile data analysis, Android architecture, Android application analysis, and mobile data collection mechanisms. The following sections briefly describe each of these areas.

2.1 Analysis of Mobile Data

Mobile phones are capable of tracking human movements, monitoring application usage, recording point of intersection, etc. Specific sets of mobile data include social relationship data, communication data, GPS trajectory data, application usage data, and sensor data. Human relationships and community interaction are the focal point of many researchers. The understanding of social relationship and community behavior can help researchers understand the spread of diseases, and crime outbreaks. This type of data also offers valuable information for business marketers.

Dong *et al.* [8], from MIT have attempted to model the co-evolution of behaviors and social relationships using mobile phone data. The research was conducted on seventy residents of an undergraduate resident hall at MIT. The data were collected through Windows Mobile devices. The data collected were self-reported surveys that were designed by experts in political sciences and medicine, and proximity and location data that were recorded from cell-phone sensors. Datasets from the survey included how residents were socializing on Facebook, Twitter, and blogs. The datasets from the sensors also included communication patterns that indicated who

called whom, and who sent short messages to whom. Figure 2.1 is a graphical representation of the interaction of students in the MIT community.

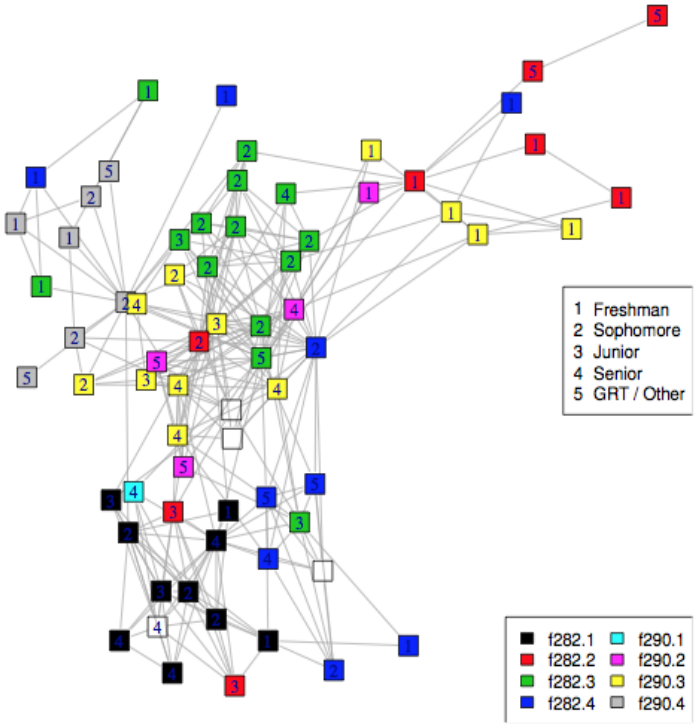

Figure 2.1: Interaction of students in the MIT community (figure extracted from Dong *et al*, [8]).

As shown in Figure 2.1 the different living sectors are represented with different colors. The numbers corresponds to the year level of each student. This also demonstrated that subjects in the dormitory formed clusters of relationships by their dormitory sectors and their years in school [8].

On analysis related to communication data, Huhtala *et al.* [21] attempted at modeling personal communication patterns by developing a mobile application. Their approach was to collect logged calls, short message service (SMS), and multimedia messaging serivce (MMS). After the data were gathered, they categorized the data as positive and negative according to the user's actions with the incoming communication attempts. A communication was deemed negative if the incoming call was declined, not answered, or did not reply to received message. The result was then put into a timeline

to display the positive and negative activity of the user. As of 2010, this research was still an ongoing.

On GPS trajectory data, in the research paper Wei *et al.* [22], mobile was used in the route inference framework. The interests were the movement routes of humans, animals, hurricanes, and vehicles. The focus was the location information along with a timestamp variable. The researchers analysis was based on check-in activities, photo uploads, and activity sharing: basically anything that has a location and timestamp attribute associated with it. This type of data has uncertainties associated with it due to the low frequencies caused by certain application features and energy saving modes [22]. The uncertainties were then used to construct popular routes without knowing road network information.

Yuan *et al.* [16] is another case of utilizing GPS trajectory data. Specifically, mobile GPS was used to determine human mobility models. Human trajectories were determined based on cell-tower traces in a cellular network, trajectory driving routes, or posts that consists of geo tweets and geotagged photos, as well as check-ins. This type of data was determined to have a strong correlation with the traveling behavior of people [16]. The researchers used this dataset to determine when people arrived at and left from a specific region. The data points were also used to determine where people were coming from and going to. This dataset was then combined with POIs to model the urban dynamics. Specifically, the topic-based method was used to analyze the data. For example, the geographic regions are seen as documents. The functions that are associated with each region are seen as topics. The human mobility gathered from mobile data is seen as words. Lastly, the POIs located in each region are seen as metadata [16]. This model was put in place to analyze the urban dynamics in Beijing.

For application usage data, in the research paper Shi *et al.* [18], mobile data were used to enhance the recommendation algorithms that Google Play and Netflix have been using for ages. GetJar is a mobile application based company. They provide applications to users of all platforms. GetJar takes advantage of the application usage data collected on mobile phones to make recommendations to users. Depending on the recent application usage of a personal user, GetJar personalized the application recommendation based

on that [18].

Aside from all the datasets discussed, there are also sensor datasets. Mobile devices are equipped with several advanced sensors to enhance the user experience. In the area of data collection, these sensors could be easily accessed and used to collect valuable information about the user. Kwapisz *et al* [20], explored the potential of using a cell phone accelerometer to detect the type of activity that the user had performed. In this particular approach they utilized the fact that accelerometers can track movement in the X, Y, and Z axes. The data were collected directly from files stored on the phones via a USB connection. The collected accelerometer values were then translated to useful data sets. The data sets contained information such as average acceleration for each axis, standard deviation for each axis, average absolute difference for each axis, average resultant acceleration, time between peaks of each axis, and range of value in each axis. The collected data were then plotted. By analyzing the different patterns and frequencies of various activities, it was then quickly realized which activity the user was performing at that time.

This body of related work demonstrates some of the many possible uses for collected mobile data. These types of analysis are based on explicit data collection through some sort of human interaction, whether it is mindful interaction or oblivious interaction. The work discussed very much focused on human mobility analysis. However, there is another side of analysis that has not been explored very much. This thesis work focuses on the implicit data collection (data collection that are completed without user awareness) and analysis of background services data. Specifically, the work concerns access and activity frequencies to privacy sensitive resources on the phone. This specific set of data may help researchers understand potential security loopholes as well as to better understand specific application.

2.2 Android Architecture

Android is a very popular modern open mobile platform. In just the third quarter of 2013, 81.3 percent of the smartphones shipped were based on the Android operating system [13]. This open source platform is composed

of three building blocks: device hardware, Android operating system, and Android application runtime [1]. The device hardware contains the hardware configurations to target specific devices such as smart phones, tablets, etc. The Android operating system is responsible for managing all device resources as camera, GPS, Bluetooth, telephony function, network connections, etc. Lastly, the Android application runtime is the environment in which both Dalvik and native applications run.

In terms of software, Android has a very defined architecture. In Figure 2.2, each layer and its responsibilities of the software stack are displayed.

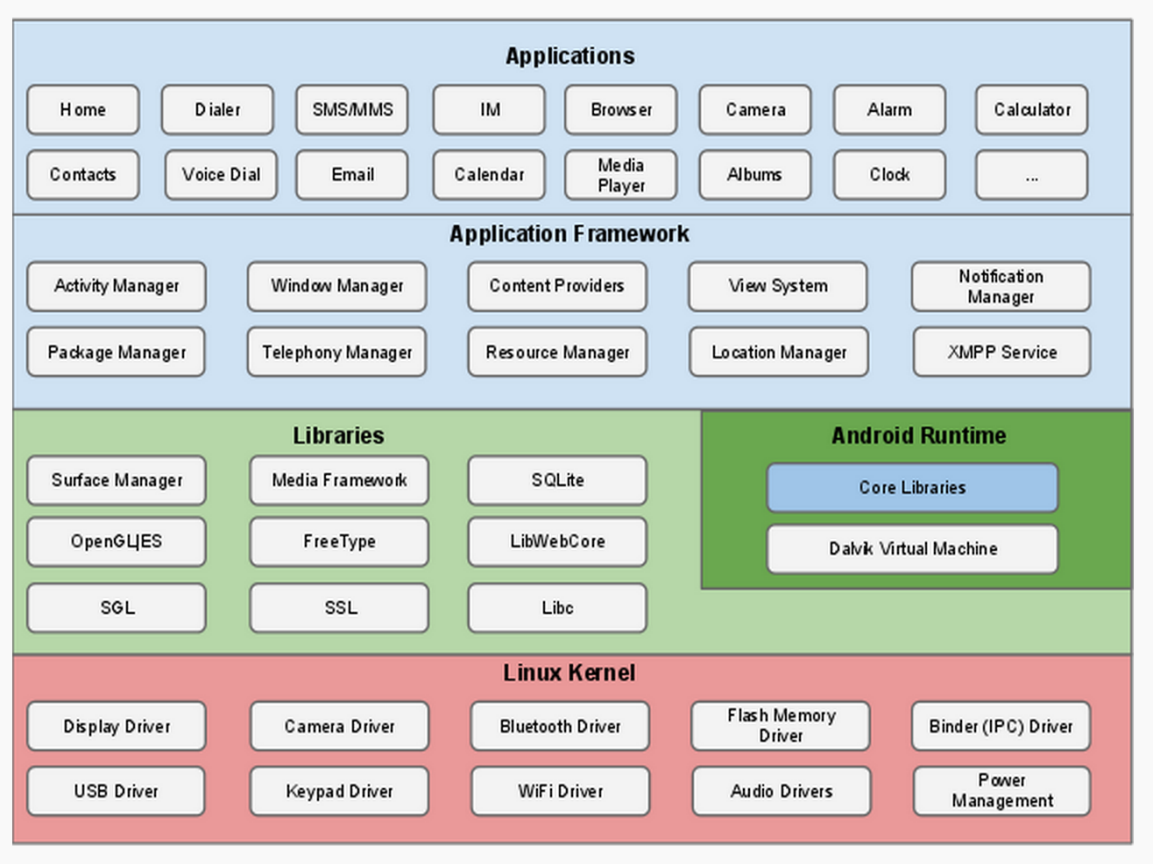

Figure 2.2: Android software stack (figure extracted from Android security overview, [1]).

The idea of the software stack is that each component assumes that the components below are properly secured. The Linux security layer provides user-based permission model, process isolation, extensible mechanism for

secure inter-process communication (IPC), and the ability to remove unnecessary and potentially insecure parts of the kernel. In this thesis investigation, the application framework layer, libraries layer, and android runtime layer are modified to support the collection mechanism.

2.2.1 Android Application Sandbox

Application sandbox is based on the Linux user-based protection concept. In general terms, it is the idea of identifying and isolating application resources [1]. A unique user ID (UID) is assigned to each Android application by the Android system. Each application then runs as that user in a separate process. This prevents applications from interacting with each other by default, and it also limits the access to the operating system. The application sandbox resides in the kernel layer; this architecture allows software above the kernel as shown in Figure 2.2 to comply with the security limitations.

This application sandbox architecture makes collection mechanism implementation impossible in the application layer as seen in Figure 2.2. Since the focus of the collection mechanism is to monitor the accesses of third-party applications to privacy sensitive resources, modification to the Android software stack was necessary to support the level of Android security breach.

2.3 Android Application Analysis

In terms of Android application analysis, there has been little work. Enck *et al.* [27], addressed the weaknesses of smartphone operating systems in terms of giving users adequate control over and visibility into how third-party applications use their private data. The approach that was taken was to create a customized ROM to allow tracking of the user sensitive information as it traverses through the phone and eventually leaves. The major concept that was exposed was the idea of labeling data from privacy-sensitive sources and transitively applying labels as sensitive data propagates through program variables, files, and inter-process messages. This research was very successful. It was tested on 30 popular third-party Android applications and

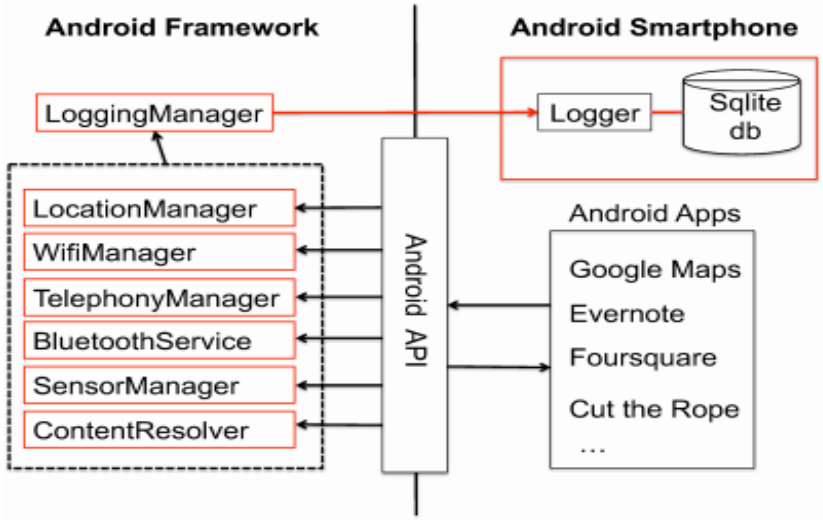

Figure 2.3: AppWindow architecture structure (figure extracted from Zhang *at el*,[11]).

68 cases of potential misuse of users' private information was found across 20 applications.

Wei *et al.* [12] is another research effor to assess and monitor applications in a systematic way. This research attempted to construct a profile for various third-party applications. The main approach is divided into four different layers: static specification, user interactions, operating system, and network. In the static layer, the APK (Android application package) was analyzed. In the user interaction layer, user generated events were recorded and logged with tools such as adb and logcat. For the operating system layer, system calls were monitored to observe operating system activity. In the network layer, network traffic was analyzed through the logging of data packets. In total, 27 free and paid Android applications were evaluated. Discrepancies between the application specification and application execution were found. Many cost-effective but comprehensive application profiles were formed.

For improving transparency for smartphone applications, Zhang *et al* [11], has created a tool to assess privacy risks involved with mobile applications. The team developed a framework to qualitatively assess and quantitatively measure the intrusiveness of smartphone applications. The

assessment was conducted based on the access behaviors of applications. Their overall system was divided into two parts: Privacy Fingerprint and Intrusiveness Score. The AppWindow developed for the Privacy Fingerprint component is shown in Figure 2.3. The basic idea was that the methods of each accessed sensitive information methods are monitored. Whenever specific methods are called, the event is logged. Intrusive scores algorithms are then applied to understand the risks involved.

This thesis work contributes to application analysis. It incorporates and extends the work done with TaintDroid [27]. The enhancements lie in the addition of privacy sensitive resource action listeners. These listeners augment the sensitive resource access dataset by providing sequential access and specific actions.

2.4 Data Collection Mechanisms

For collecting information on a mobile device, there are many approaches. In their research paper Striegel *et al.* [6], the question how the digital world (Facebook, SMS, etc) impacts how friends are made and kept was explored. A NetSense Infrastructure built to extract useful information from the mobile device and pass it on for storage on an external server. The overall structure is displayed in Figure 2.4.

The structure was very logical. There were essentially two components: data gathering agent and database server. The data gathering agent was a pre-installed application on each of the Nokia phone for experimentation. The pre-installed agent was executed at the user level. The types of data that were collected by the agent were application usage, device usage, location, communication, proximity of other users, device state, and network environments information. Since the collection was not event triggered, the frequency of collection was based on a periodic basis ranging from hourly to daily. Once the data were collected, they were first temporarily stored in a local SQLite database on the mobile device. After, the data were then sent to the two check-in servers. The purpose of the check-in servers was to provide fault tolerance in the event of failure [6]. After the check-in servers, the data were then parsed and stored in a relational database.

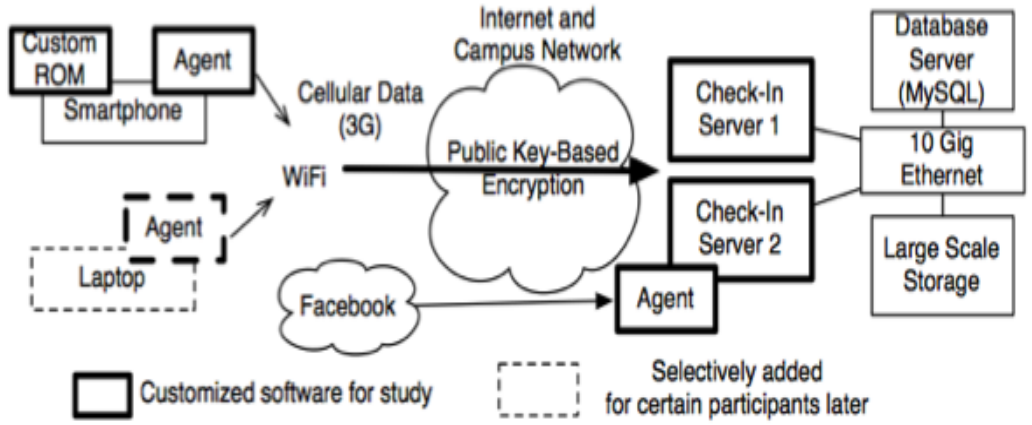

Figure 2.4: NetSense mobile data aquisition infrastructure (figure extracted from Striegel *at el*,[6]).

In Zhang *et al.* [14], a user modeling framework was developed from a multimodal mobile sensor and log data. The framework that was built covered the whole process from collecting, to processing, to analyzing, and applying data. mFingerpring framework is displayed in Figure 2.5

The framework was clearly defined in four layers. The very first layer is the mobile data collection layer. In this layer, an Android app called Easy-Track was developed based on Funf Open Sensing Framework. The app collected hard sensor data as well as application log data. The collected information was then passed to the feature computation layer. In this layer, the entropy and frequency features were then conditioned based on time and location. After, data mining techniques such as clustering and classification were carried out in the user learning layer to further massage the data. The learned insights were then applied in the application layer in areas of user profile/behavior logger, reminder and user identification, and personalization services/UI.

The related works presented very logical process and structure from collecting to storing mobile data. The two component approach is very modular and energy efficient. This thesis work follows very much of a same

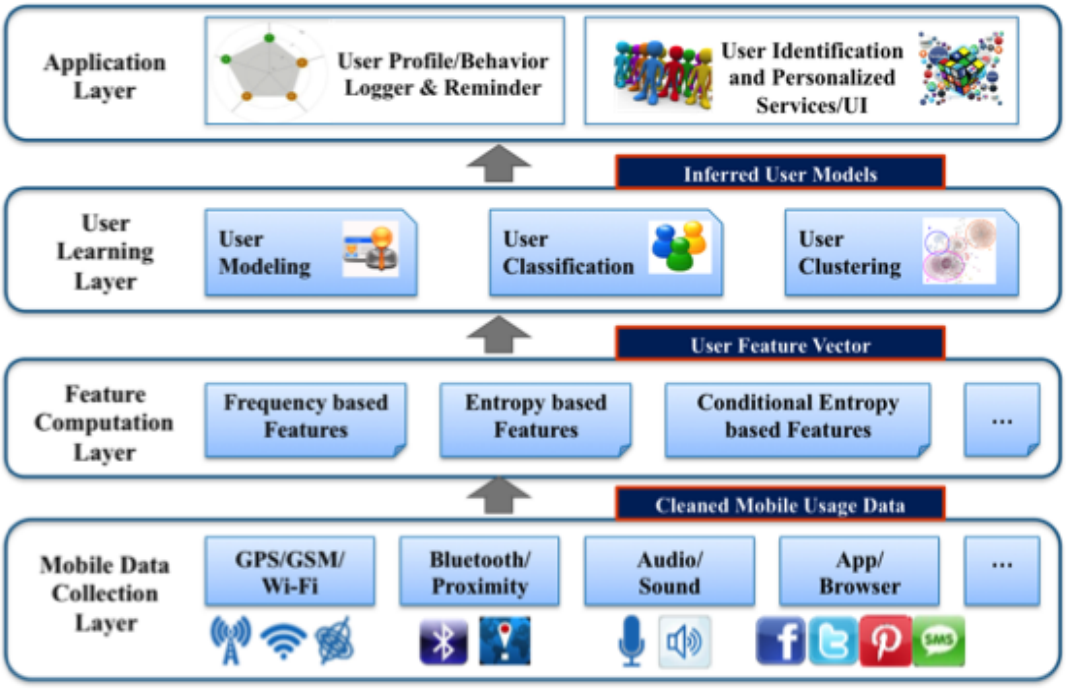

Figure 2.5: mFingerprint mobile data collection to application framework (figure extracted from Zhang *at el*,[14]).

structure. There is a collection agent pre-installed on the phone to monitor the communication between Android background services and sensitive resources. This agent is based on the customized ROM developed through the study done by Enck *et al.* [27]. The database storing the information is also externalized on a server. However, this thesis work introduces a third component: the collection client. The collection client is also pre-installed on the phone. This client handles the communication between the database and the collection agent. By adding the collection client, it reduces any bottleneck that the collection agent might have trying to reach a connection with the database server.

Chapter 3

Methodology

The lack of understanding of application and sensitive resource communication presents a need to collect dynamic and timestamp based data on third-party applications' access to various available privacy sensitive resources. To approach this system wide sensitive resource monitoring system, a dynamic, sequence-based system must be implemented to track the various sensitive resources and store the resulting data. To design a dynamic real time privacy sensitive resources monitoring system there are many things to consider.

1. System must be able to listen to the communication between available privacy sensitive resources and other third-party applications.
2. System must be able to track the flow of sensitive resources data.
3. The collection must run passively in the background to prevent interference with current running applications.
4. The collected data must be stored in a scalable database to support future expansion.

The later sections of this chapter cover how each of these requirements was satisfied. First, an overall system design analysis discussed to assess the pros and cons of each design concept. Secondly, a high level overall view of the chosen design explained. After, each module of the overall system design discussed in details.

3.1 Overall System Design Analysis

In order to accomplish these tasks, a dynamic system wide time based sensitive resource tracking system implemented. The design consideration was divided into two major focuses: the collection module and the server module.

3.1.1 Collection Module Analysis

The collection module is the most important and crucial consideration of this thesis investigation. For the collection module design, the main requirement is that the module must be able to observe and record the communications of privacy sensitive resources. At a first glance, the very logical way of tracking system wide communication between third-party applications and sensitive resources was to design an application to monitor the processes of other applications. This type of design calls for application process and resource monitoring. With the current Android sandbox security architecture, this type of activity was marked as a type of security breach. Figure 3.1 displays the Android security structure for user sensitive data access.

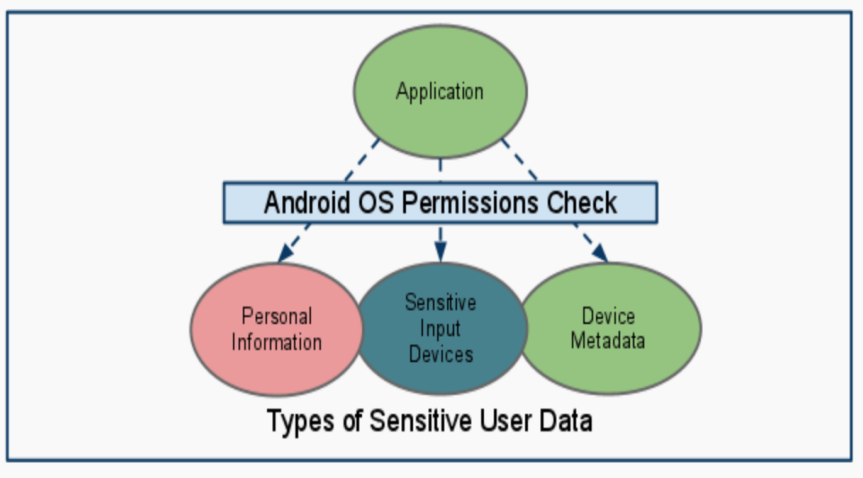

Figure 3.1: Android security structure for user sensitive data access (figure extracted from [1]).

The idea of the sandbox architecture is to isolate each application processes from the other to ensure that there is no memory, CPU, and device (e.g. telephony, GPS, Bluetooth) hogging [1]. The sandbox structure also protects one applications private data from the others. Due to this security structure, a third-party sensitive resources monitoring application was not feasible.

Another examined approach was to utilize the Android debug bridge (ADB) and the Android logging system. The Android system supports a system wide logging facility. This facility allows for the logging of applications and system components. The logging system consists of a kernel driver and buffer for storing log messages, language classes for making and accessing log messages, a standalone program for viewing log messages (logcat), and it has the ability to filter log messages from the host machine (via Eclipse or DDMS) [2]. The system also supports four different log buffers to enable logging for different parts of the Android system. The four log buffers are: main, events, radio, and system [2]. The main buffer is responsible for main application logging. The events buffer is used to track system even information. The radio buffer is used to log radio and phone based activities. Finally, the system buffer is used to log low-level system and debug messages.

After some analysis, it was realized that this logging system is only useful to a certain extend. For the logs to contain communication data between third-party applications and sensitive resources, third party applications had to implement log triggers to log those messages. This required a modification to all of the third-party applications' source code. Aside from the modification, the already logged messages are also not in a standard format. The message part of the log contained whatever the user or system wanted, thus made analyzing the collected dataset extreme difficult. With these two limitations, the logging design of the collection module would be impractical.

The final collection module design that was analyzed was the idea of taint tracking, which was successfully demonstrated in the TaintDroid research [27]. The idea was to modify the ROM to enable tagging of privacy sensitive resources. The tagging allows the tracking of sensitive data flow

throughout the device to when the sensitive information leaves the phone. This concept was successful in tracking when changes were made to the sensitive data source as well as when sensitive information left the phone. This type of tracking lacked details in specific actions performed by each application processes. For example, all the internal details of actions performed before the information left the phone were ignored. This limitation required the implementation of listeners to track specific internal sensitive resource activities. The design is a system of taint tracking combined with privacy sensitive resource listeners. This provides a more comprehensive and sequence based sensitive resources communication dataset.

3.1.2 Server Module Analysis

The purpose of the server module is to handle the collected data in an efficient way that could make future analysis easier. For the scope of this thesis investigation, the server module had to be able to parse data strings and store them in a database. The communication method between the client and the server was hypertext transfer protocol (HTTP). This protocol supports two different request methods: GET and POST. The GET method is a simple and insecure way of requesting data from specified resources. GET requests can be cached and bookmarked, and they have length restriction. On the contrary, the POST request is never cached, does not remain in the browser history, and has no data length restrictions. Due to the sensitivity and variable length of the data being collected, the HTTP POST request method was chosen for communication between the client and server module. As for the database, MySQL database was chosen. Since the communication data between third-party applications and sensitive resources are all very standard and structured, the MySQL database satisfied the need. The structured MySQL database allowed to easy data analysis as well.

3.2 Data Collection Components

The purpose of this overall system design is to enable the tracking of actions taken by third-party applications that are related to sensitive resources. As

described in the overall system design analysis section, many design concepts were viewed and evaluated against system requirements.

To satisfy the requirements mentioned previously, the overall system is composed of three components: data collection module, collection client module, and collection server module. Figure 3.2, is a high level overview of the overall system components and process flow.

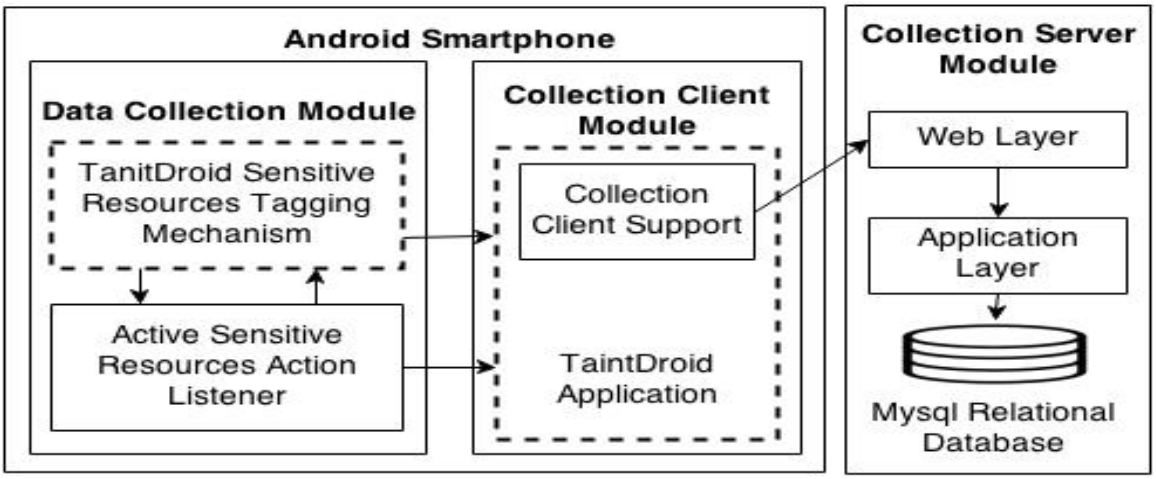

Figure 3.2: Overall privacy sensitive resource monitoring system design.

The design is based on the Android operating system. As seen in Figure 3.2, the design is divided into two major implementations: the Android smart phone implementation and the server side implementation. The Android smart phone implementation is then further divided into two different modules: the data collection module and the collection client module. Further division is based on the Android architecture structure and process flow. The data collection module resides throughout application framework, libraries, and Android runtime layer as could be observed in Figure 2.2. The collection client module is implemented as an Android application and resides in the application layer.

The data collection module is designed based on the TaintDroid system [27]. The module utilizes the idea of tainting the data to track sensitive information flow. The TaintDroid tainting component is run in parallel with active sensitive resources listeners to track resource specific actions. The combined structure allows the system to listen to communication between

available sensitive resources and other third-party applications. The incorporation of TaintDroid also allows the system to tag sensitive resources and track the flow of data from them throughout the system. Since the sensitive resources listeners are placed in the application framework layer, this enables the collection to run passively in the background to prevent interference with other running applications.

The collection client module is also designed based on the TaintDroid-Notify application [27] with the addition of client and server communication support. The client module defines the dataset structure and prepares the data to be sent to an external server. The communication between client and server is established by the collection client support component as seen in Figure 3.2. The communication protocol that is used is HTTP POST request. Since the application spawnned Android services to actively collect and send information, the collection process was able to run passively in the background.

On the sever side implementation, the design was a simple one. The goal was to securely receive the data and establish an interface between the server application layer and the implemented MySQL database. The client module is composed of three components as described in Figure 3.2. The components are web layer, application layer, and MySQL relational database. The web layer handles incoming server communication. The application layer is responsible for establish database connection. Finally, the database is used to house all the incoming data. This structure allows for secure data transfer and storage due to the security check in the application layer. This structure could be copied and more databases could be added to support future expansion.

The process flow of the overall system can be observed in Figure 3.2. The process is resources access event driven. The resource action listeners actively records sensetive resource based events. These events are tagged with the TaintDroid tagging system [27]. The events are then passed on to the client module for data formatting and preparing for send off. The events are formatted in key value pairs. Then, the collection client support component establishes a secure connection with the server module, and sends the event through HTTP POST request. The request first hits the web layer,

then it passes to the application layer. In the application layer, the data is further formatted to meet the database schema. Once the application layer establishes a secure connection, the event is then injected into the database. Users can then observe the results in the database.

The later subsections describe each module implementation in detail. The data collection module is discussed first, and then the client module. Finally, the section concludes with the server module.

3.2.1 Data Collection Module

The most crucial part of the overall privacy sensitive resource monitoring system is the data collection module. The collection module is responsible for monitoring sensitive information flow as well as actions taken by each sensitive resource. The overall data collection module is described in Figure 3.3.

The design of the collection module is composed of two specific implementation components: TaintDroid [27], and sensitive resource listeners. TaintDroid is an information-flow tracking system that was originally developed by Enck *et al.* [27]. The system automatically labels (taints) data from privacy-sensitive sources. The specific levels of tracking are variable-level, method-level, and file-level. As sensitive information flows through the phone, transitively, the labels are applied and propagated. The current collection module incorporates TaintDroid by utilizing the TaintDroid logging and tagging system. As shown in Figure 3.3, the colored areas are modification done by TaintDroid. To support tracking of resource usage within Android, custom listener components were implemented in application framework layer. The specific components in the application framework layer are the implemented resource action listeners. These listeners track specific actions that could be performed by each sensitive resource.

With the incorporation of the TainDroid system, there are a total of eleven sources that are being tracked, and the identified sensitive sources are:

- Location
- Contacts

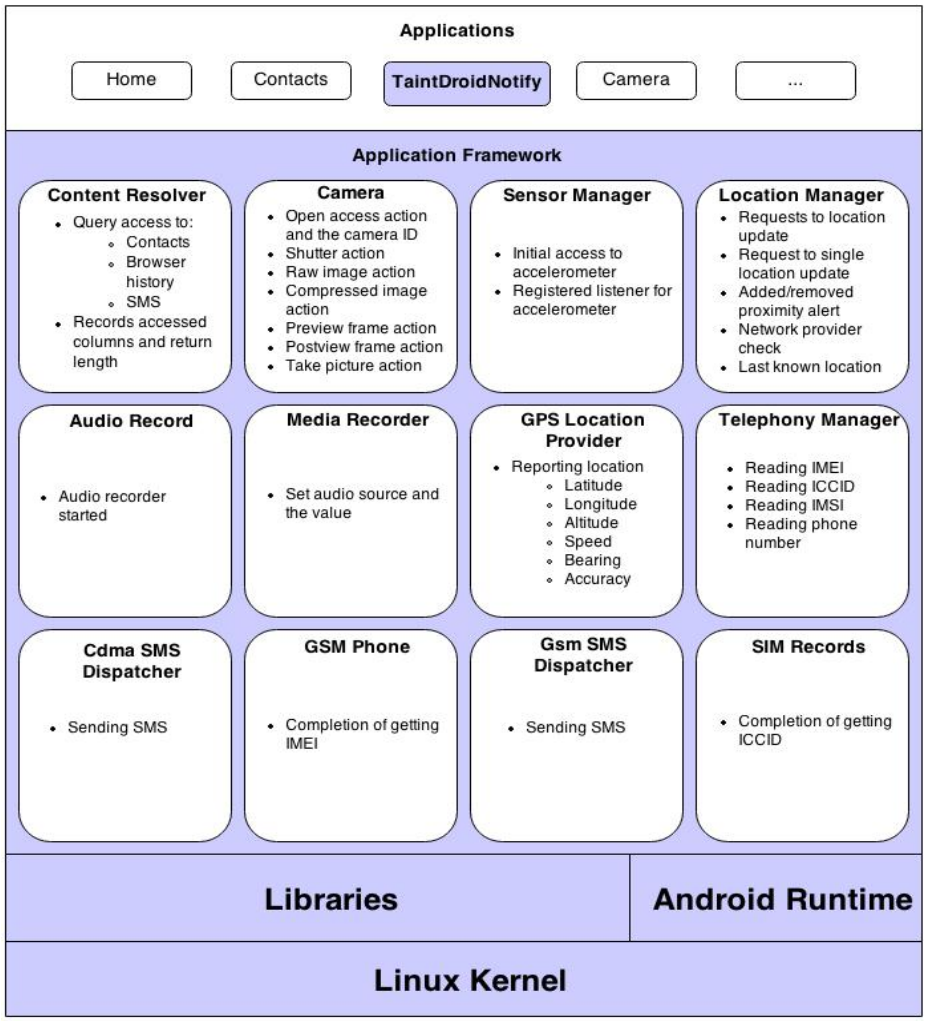

Figure 3.3: Overall data collection module design.

- Microphone
- Phone Number
- Camera
- Accelerometer
- SMS
- IMEI
- IMSI

- ICCID
- Browser History

In Figure 3.3, the components in the application framework layer corresponds to the specific listeners needed to track the eleven sources listed. There are two types of listeners, one that is responsible only for logging specific actions, and one that is responsible for logging the action as well as returning data gathered from that action. The listeners are implemented through the TaintDroid logging system. For only action specific listeners, the resource label and the specific actions are logged in text format. For example, if a third-party process is using the microphone, the taint tag "TAINT_MIC" will be logged as will as the action "start recording". For action listeners with returning value, the returning value will first be retrieved. The retrieved value is then logged with the tag information as well as the action performed.

Figure 3.3, displays all the components that are necessary to track all the eleven sensitive resources. First, for the location resource, there are multiple things being tracked. The location listeners are implemented in two components: location manager and GPS location provider. Table 3.1, displays the responsible components and various action listeners.

Table 3.1: Location listeners

Sensitive Resources	Framework Component	Listeners
Location	• Location Manager • GPS Location Provider	• Location update • Proximity alert • Network provider check • Reporting location

The location update listener listens to third-party requests for location update as well as request for last known location. This listener simply records that the specific event occurred. The proximity alert listener listens to when a proximity request is added or removed. This listener also reports

the proximity target. The network provider check listener simply checks if the device has the network provider active. The reporting location listener listens to when an actual location read occurred. This listener reports the event and location data: latitude, longitude, altitude, speed, bearing, and accuracy.

Table 3.2: Contacts listeners

Sensitive Resources	Framework Component	Listeners
Contacts	• Content Resolver	• Contact read

The contact resource is managed by the content resolver component. Table 3.2 briefly describes the component and listener. The listener that is responsible for monitoring contact activity is the contact read listener. This listener listens queries to contacts. Aside from recording the event, the accessed column names and the result length is also recorded.

Table 3.3: Microphone listeners

Sensitive Resources	Framework Component	Listeners
Microphone	• Media Recorder • Audio Record	• Audio recorder initialization • Audio source and status

For microphone use, there are two components that are responsible for actions. They are Media Recorder and Audio Record. The audio source and status listener is attached to the Media Recorder component. This listener is responsible for recording the event when an audio source is added. This listener also records the added audio source ID. The audio recorder initialization listener is implemented in Audio Record component. This listener listens to the microphone initialization event and records the event.

Table 3.4, shows the listeners and components responsible for device specific resources such as phone number, IMEI, IMSI, and ICCID. The main component that is responsible is the Telephony Manager. For each resource,

Table 3.4: Device information listeners

Sensitive Resources	Framework Component	Listeners
Phone Number	• Telephony Manager	• Reading phone number
IMEI	• Telephony Manager • GSM Phone	• Reading IMEI number • Reading IMEI number complete
IMSI	• Telephony Manager	• Reading IMSI number
ICCID	• Telephony Manager • SIM Records	• Reading ICCID number • Reading ICCID number complete

there is a listener attached in the Telephony Manager component to track the reading event and the resulting accessed value. For IMEI, there is a second listener attached to the GSM Phone component to track the IMEI reading completion event. The ICCID reading completion event follows the same structure as the IMEI reading completion event except that it is implemented in the SIM Records component.

Another resource that is monitored is the camera. The camera utilizes the Camera component in the Android framework layer. This is shown in Table 3.5. There are seven specific listeners attached. The camera initialization listener is attached to record the event when the camera is initialized by a third-party application. The other listeners track the specific process that the camera goes through: shutter action, raw image processing, image compression, preview frame and post view frame. Lastly, the take picture action listener records the action of physically taking a picture.

For monitoring the accelerometer resource, listeners are attached to the

Table 3.5: Camera listeners

Sensitive Resources	Framework Component	Listeners
Camera	• Camera	• Camera initialization • Camera shutter action • Camera raw image action • Camera compressed image action • Camera preview frame action • Camera post view frame action • Camera take picture action

Sensor Manager component. Table 3.6 lists the listeners. The main goal here is to monitor the accelerometer initialization and the register of callback listeners. Thus, the initialization listener is attached to the accelerometer initialization process to record the event. For accelerometer, there are many ways to register listeners; therefore, a separate listener is attached to each register listener method to listen to all registering events.

Table 3.7, displays the framework components and listeners responsible for monitoring the SMS resource. The three components involved are CDMA SMS Dispatcher, GSM SMS Dispatcher, and Content Resolver. In

Table 3.6: Accelerometer listeners

Sensitive Resources	Framework Component	Listeners
Accelerometer	• Sensor Manager	• Accelerometer initialization • Registered listener for accelerometer

Table 3.7: SMS listeners

Sensitive Resources	Framework Component	Listeners
SMS	• Cdma SMS Dispatcher • Gsm SMS Dispatcher • Content Resolver	• Reading SMS • Sending SMS

CDMA and GSM SMS Dispatcher, the sending SMS listener is attached. Here, the listener records the event when SMS message were sent out. The reading SMS listener is attached to the Content Resolver component. This listener records the query access to the SMS message table. Along with the event action, the columns queried and the returning length are recorded as well.

Table 3.8: Browser history listeners

Sensitive Resources	Framework Component	Listeners
Browser History	• Content Resolver	• Reading browser history

The final resource is browser history. This resource is monitored in the same way as contacts. Table 3.8, shows the specifics. The component involved is Content Resolver. The listener attached is the browser history reading listener. This listener is attached to the query access for browser history. Whenever a browser history query is made, the queried columns and returning length are recorded.

Each of the twelve components in the application framework layer and their responsible listeners, shown in Figure 3.3, actively integrates with the already existing TaintDroid system to dynamically and passively monitor the named eleven sensitive resources. This concludes the design and implementation discussion for the data collection module. Next, the collection client module will be discussed.

3.2.2 Collection Client Module

The collection client module is the component that communicates between the data collection module and the collection server module. Essentially, the client module is the messenger. As can be observed in Figure 3.2, the collection client module resides on the mobile device. This component captures the listener logs, formats the events, and then sends the formatted data to the server module. This module is developed as a third-party application called TaintDroidNotify. TaintDroidNotify was originally developed by Ench *et al.* [27]. Modifications were made to support event formatting and server communication. This package resides in the application layer of the Android software stack as seen if Figure 3.3. Specific components and process flow of the client module is shown in Figure 3.4.

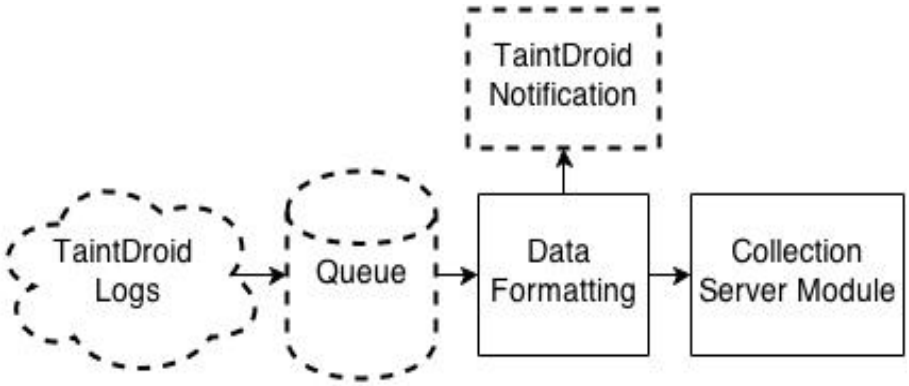

Figure 3.4: Overall data collection client module design.

For the client module, there are essentially two components. The first component is the queue. This queue is driven by two active Android services. One of the service runs passively in the background to collect all of the incoming resource access events. The collected events are then inserted into the queue. The purpose of the queue is to reduce potential bottlenecks from the formatting process, and to make sure all events are captured. The second service runs passively in the background as well. This service takes the logs out of the queue one at a time. Once the logs are taken out, they go through a data formatting process. This process is done in the data formatting component. In the component, information such as application name,

IP address, taint tag, timestamp, and data messages are fetched. This fetched information is then put into a key value format. The formatted message is then sent to the phone device as a notification message or/and sent to the collection server for storage. An example of the data format and the notification message is shown in Figure 3.5.

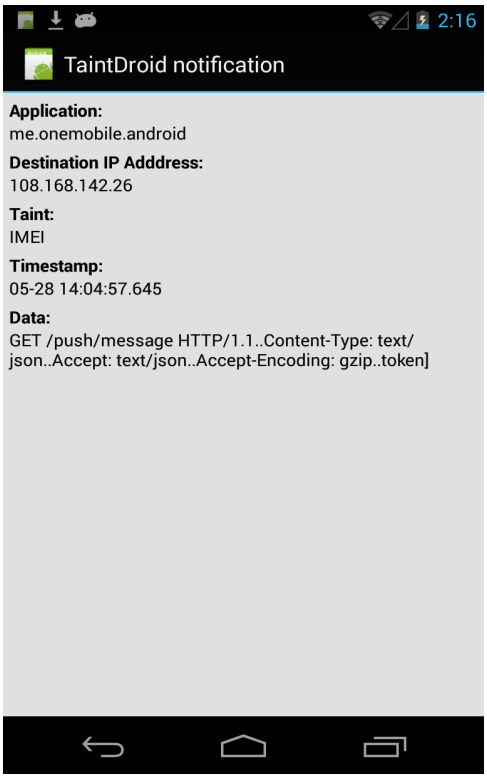

Figure 3.5: Example TaintDroidNotify notification.

The method of communication to collection server module is through HTTP POST request. This is done though the use of Android HttpClient. First the HttpClient is initialized. Then a POST request is made to the server module URL. A name value paired list is then used to store the event data. After, the name value paired list is added to the POST request. Finally, the HttpClient attempts the transfer of data by executing the POST request. If the POST was successful, HttpResponse then receives a success response message.

In summary, the collection client module enabled passive retrieving and sending of sensitive resource access events. It allowed the communication

between the collection module and the server module to carry on seamlessly. The next section will discuss the implementation details of the collection server module.

3.2.3 Collection Server Module

The final component of the privacy sensitive resource monitoring system is the server module. The server module is responsible for receiving incoming message strings, processing the message strings, and inserting them into the database. Figure 3.6, explains the overall server module components and process flow.

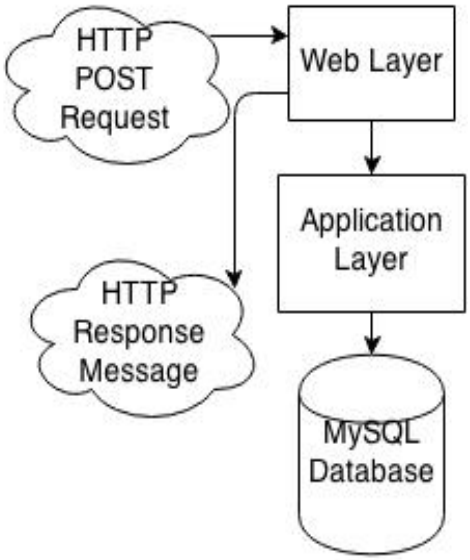

Figure 3.6: Overview of collection server module design and process flow.

There are three main components for the server module as seen in Figure 3.6. The tasks of the server module are to receive HTTP POST requests and insert them into the database. Thus, an Apache web service had to be installed to handle incoming HTTP requests. The web service is composed of the web layer and the application layer. The web layer is the outside facing interface. All the webpages and actual displays reside in this layer. Past the web layer is the application layer. The application layer handles the communication between the web layer and the database. For the purpose of

this server module, the application layer is written in PHP to interface with the MySQL database. The application layer establishes connection with the MySQL database using user and password security checks. This layer also handles the parsing of incoming POST requests. The POST messages are formatted to fit into the MySQL database schema. The schema is shown in Figure 3.7.

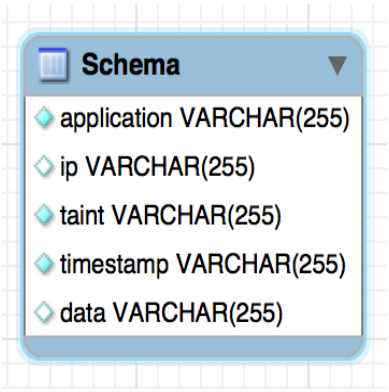

Figure 3.7: MySQL database schema.

Finally, the schema defines the MySQL database. The schema follows a very structured layout. There is only one table in the database, since all income messages are of the same type. There are a total of five columns in the table. These five columns correspond to the key value structure setup in the collection client module. The application column stores the application name of each triggered event. The IP column stores the IP addresses of the triggered events. The taint column is the column specifying the type of resources that were accessed. Timestamp column records the exact moment when the resource event was triggered. Finally, the data column records extra data passed in by the implemented listeners.

The process flow of this module is a rather simple one. The web layer receives HTTP POST requests, then the request is passed on to the application layer and a response message is sent back to the client module. The client module checks for valid message body then attempts connection with the MySQL database. Once a connection is established, the message body is put into the MySQL schema format and inserted into the specified table. Finally, information stored in the MySQL database can be queried and

analyzed.

This concludes the overall design and implementation of the privacy sensitive resources monitoring system. The next section discusses the test application design and final verification against the requirements.

3.3 Test Application Design and Implementation

A test application was created to verify the overall functionality of the system. The specific focus of the test application is to mimic third-party application processes and test the resource listeners implemented in the Android application framework layer. The overall design concept is based on the onClickListener object. For each defined third-party application process, an onClickListener object is created to trigger that specific action. The resulting application that was created is named SensorTrigger, and a screenshot of the application is shown in Figure 3.8.

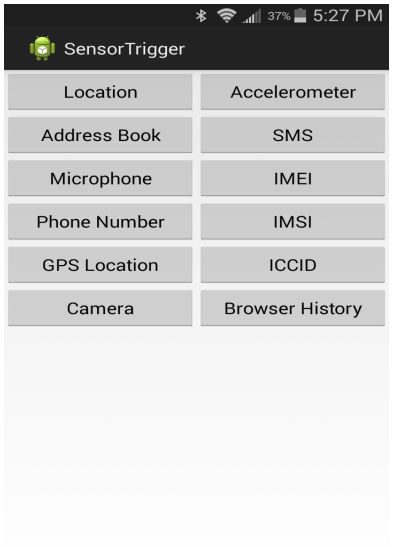

Figure 3.8: SensorTrigger application display.

The twelve buttons displayed in Figure 3.8 corresponds to the eleven sensitive resources and their listeners. First, for location resource testing, two separate onClickListener were created. One was used to test network based

Table 3.9: Location based resourse test

Sensitive Resources	Tested Listeners	SensorTrigger Buttons	SensorTrigger Actions
Location	• Location update • Network Provider check • Reporting location	• Location • GPS Location	• Request location update • Request get last known location • Request GPS/Network check

location reporting and the other was used to test GPS based location reporting. These onClicklisteners trigger separate background location reporting services. Both services are triggered to be active for twenty-five seconds to allow their services to complete the location reporting cycle. A complete match between location resource listeners tested and SensorTrigger actions performed are displayed in Table 3.9.

Next, for triggering events by contacts, browser history, and SMS related resources, three separate onClickListeners were created. Table 3.10 displays the complete match between contacts, browser history, and SMS resource listeners tested and SensorTrigger actions performed. These triggers are implemented using the same process. First, request columns are defined. Then a Cursor object is created to query the specific resource.

The next resource tested was the microphone. For the microphone, an onClickListener is created to perform the recording action. First, a MediaRecorder object is created. With this MediaRecorder object, the audio source, output format, output file name, and audio encoder are set. After the microphone is set to start and record for five seconds. After the five seconds, the microphone is turned off and the resource is released. Table 3.11, shows the specific actions performed as well as the listeners tested.

The accelerometer sensor was also incorporated into the resource test.

Table 3.10: Contacts, browser resource, sms based resourse test

Sensitive Resources	Tested Listeners	SensorTrigger Buttons	SensorTrigger Actions
Contacts	• Contact read	• Address Book	• Read contacts id and name
Browser History	• Browser history read	• Browser History	• Read browser history title and url
SMS	• SMS read	• SMS	• Read SMS id, address and body

The implementation of the accelerometer resource triggers follows a similar structure as the location based trigger implementation. Table 3.12, displays the accelerometer based resource actions and listeners. For implementation, first, an onClickListener is created for the accelerometer trigger. Then, a background service is created to perform accelerometer-based actions. Within the service, SensorManager is created with accelerometer added as the default sensor. Accelerometer listeners are also registered to retrieve accelerometer readings.

Table 3.11: Microphone based resource test

Sensitive Resources	Tested Listeners	SensorTrigger Buttons	SensorTrigger Actions
Microphone	• Audio recorder initialization • Audio source and status	• Microphone	• Initialize audio recorder • Perform recording

Table 3.12: Accelerometer based resource test

Sensitive Resources	Tested Listeners	SensorTrigger Buttons	SensorTrigger Actions
Accelerometer	• Accelerometer initialization • Registered listener for accelerometer	• Accelerometer	• Initialize accelerometer • Register listener for sensor change events

The camera resource test implementation was a little different from the rest of the resource test implementations. The camera resource test utilizes the Android intent action. The intent triggers the Camera application on the Android device. With the triggered Camera application, user has to perform the listener-based actions to evaluate the results. Specifically, the user is required to take a picture using the Camera application on the Android device. Table 3.13, displays camera resource based listeners and specific actions taken by the SensorTrigger application.

Finally, for resources such as IMEI, IMSI, ICCID, and phone number, since they all utilize the same type of resource their trigger implementations are all similar. They all use the TelephonyManager class, thus a onClick-Listener and a TelephonyManager object are created for each resource. The specific tasks performed are simply read access. This is shown in Table 3.14.

The SensorTrigger application is used to trigger all of the implemented resource listeners. The triggered events are captured by the privacy sensitive resource monitoring system implemented. The results were then verified in the database table of the client server module. The SensorTrigger application successfully tested the functionality of the overall resource monitoring system.

Table 3.13: Camera based resource test

Sensitive Resources	Tested Listeners	SensorTrigger Buttons	SensorTrigger Actions
Camera	• Camera initialization • Camera shutter action • Camera raw image action • Camera compressed image action • Camera preview frame action • Camera post view frame action • Camera take picture action	• Camera	• Initialize camera intent • Set output file location

Table 3.14: IMEI, IMSI, Phone number, and ICCID based resource test

Sensitive Resources	Tested Listeners	SensorTrigger Buttons	SensorTrigger Actions
IMEI	• Reading IMEI number • Reading IMEI number complete	• IMEI	• Read IMEI
IMSI	• Reading IMSI number	• IMSI	• Read IMSI
Phone Number	• Reading phone number	• Phone Number	• Read phone number
ICCID	• Reading ICCID number • Reading ICCID number complete	• ICCID	• Read ICCID

This concludes the design and implementation of the overall privacy sensitive resource monitoring system as well as the test framework. The overall design satisfied all of the requirements previously defined. The implemented resource listeners in the application framework enabled the monitoring of communication between available privacy sensitive resources and other third-party applications. The combined integration of TaintDroid and active resource listeners allowed sensitive resources to be tracked throughout the mobile device. The utilization of the Android background service in the collection client module allowed the system to retrieve TaintDroid log

information passively without interfering with currently running applications. Finally, the addition of the collection server module and the database setup gave the system the flexibility to scale and expand. The next chapter explores the data collected.

Chapter 4

Results and Discussions

The privacy sensitive resource access monitoring for Android systems is a dynamic resource action listening system. The system is capable of identifying and observing direct accesses of third-party applications to sensitive resources on the phone. The recorded events are composed of name of source application, type of resource being accessed, IP address if the accessed information was sent out, a timestamp of when the resource was accessed, and finally a message body of the specific recorded action.

Relative to the original problem statement: Given the need and benefits of background services accessing mobile resources, there are both security and privacy risks. The primary motivation behind the implementation of the overall system and data collection is to demonstrate some interesting application behaviors and the potential usage of the application behavior data collection process. Specifically, application behaviors will be observed through the analysis of sensitive resource utilization, access intensity, and sequence based analysis. Potential usage of the interesting findings will also be discussed. This section will start with experiment setup followed by analysis of results.

4.1 Experiment Setup

The overall privacy sensitive resource monitoring for Android system is implemented in Android 4.3 release 1 stock ROM (Read-only Memory). Later, the finished ROM is installed and tested on a Nexus 4. The installed applications and their categories are shown in Table 4.1. There are a total of forty-seven third-party applications installed.

Table 4.1: Categories and applications downloaded.

Categories	Applications
Tools	Flashlight, Compass, DigiHUD, Location, my Location, PowerTutor, Shazam, SmartVoice Recorder, TalkBack, Uber
Games	2048 Puzzle, Basketball Throw, Wordsearch
Productivity	Adobe AIR, Duolingo, Google Text-to-speech Engine, Google Voice, Job Search, Quickoffice, SpeechToText Notepad, Zillow
Books	Bible, Guess the Emoji Answers, Manga, Manga Rock
Finances	BofA, Chase, Square Register
News	BuzzFeed, CNN , FIFA
Shopping	Cartwheel
Entertainment	Change My Voice, Pandora, Rising Start, Spotify
Social	Facebook, LINE, PicsArt, Snapchat, Twitter
Health	iTriage, MyFitnessPal, Running, SleepBot
Personalization	Kitty Play
Weather	The Weather Channel

The idea behind picking the applications shown in Table 4.1 is to observe as many types of applications as possible. The applications downloaded are all popular applications in each of their categories displayed. The hope is that different types of applications behave in a different ways. After the applications were installed, two different data collection process were performed. The two different data collection processes were passive data collection and active data collection. The idea was to observe behavioral differences of third-party application in a passive environment as well as an active environment. The later subsections discuss the collection process setup for collecting both passive and active data.

4.1.1 Passive Data Collection

The motivation behind passive data collection is to observe third-party application behaviors in an environment without user interaction. This is to spectate and identify typical behaviors of third-party applications. The environment setup for this type of collection was simple. A four hour time frame window was chosen. The time frame window size was chosen based on observations of passive data throughout the day. From observation, it was concluded that third-party applications behave in the same way throughout the

day. Thus, the four hour time frame chosen to provide a sufficient amount of data for analysis as well as capture the essential behavior of third-party applications in a passive mode.

With the chosen time frame, the following steps were taken to initiate the collection process:

1. Turned on the Nexus 4 device with installed applications and implemented data collection system
2. Started TaintDroidNotify application to enable the collection process
3. Left the phone without further user interaction for four hours
4. Turned off the TaintDroidNotify application to end the collection process

The collected data were then examined and observed in the MySQL database setup. The next subsection will discuss the setup for active data collection.

4.1.2 Active Data Collection

The active data collection is a bit more complicated than the passive data collection. The goal of active data collection is to capture third-party application behaviors when the user is actively interacting with the mobile device. The environment setup is to mimic that of a real user in a typical day. In this experiment, no real human study subjects were used. For the setup design, there were two main considerations: time window of experimentation and types of activities. First, to make the comparable between the passive and active data, the time frame window for collection was also set to be four hours. Secondly, to best mimic a typical user in a four-hour time frame, a set of specific actions was defined and carried out.

The activities and time spent on each activity were constructed based on an article written by Ballve [7]. A study done by BI Intelligence discovered that on average, U.S. smartphone owners spend fifty-eight minutes daily on their phone. The breakdowns of each activity were also discovered. The

most intensive activity is talking, which dominated the twenty-eight percent of the fifty-eight minutes. The rest of the activities are as follows: texting (twenty-one percent), social networking (seventeen percent), web browsing (fifteen percent), gaming (nine percent) and others (ten percent) [7].

In order to mimic the behavior of a typical user in a four-hour time frame, the fifty-eight minutes used by an average user in a day had to be further divided. Intuitively speaking, out of twenty-four hour period, on average, eight hours are spent sleeping and the other eight hours are allocated to working. Essentially, most of the fifty-eight minutes spent on the phone by an average user would be spent in an eight-hour time window. The exact breakdowns of time spent on each activity is shown in Table 4.2.

The activities performed and the duration of time for this experiment are so follows:

1. Turned on the Nexus 4 device with installed applications and implemented data collection system
2. Started TaintDroidNotify application to enable the collection process
3. Four minutes (passive)
4. Used LINE application to text
5. Browsed the internet for two minutes and sent two texts
6. Eight minutes (passive)
7. Checked Facebook for five minutes and sent two texts
8. Eighteen minutes (passive)
9. Played Basketball Throw for five minutes and sent two texts
10. Nine minutes (passive)
11. Sent one text
12. Thirty two minutes (passive)

13. Read CNN for two minutes and sent two texts
14. Eight minutes (passive)
15. Checked Twitter for one minute and sent one text
16. Thirty four minutes (passive)
17. Browsed for two minutes and sent two texts
18. Seven minutes (passive)
19. Checked Twitter for two minutes and sent two texts
20. Fifty-four minutes (passive)
21. Checked Twitter for one minute and sent one text
22. Forty-seven minutes (passive)
23. Turned off the TaintDroidNotify application to end the collection process

Table 4.2: User interaction activity time breakdown.

Activity Type	Utilization Percentage	Utilization Minutes	Actual Minutes
Talk	28	16.8	0
Text	21	12.6	9
Social	17	10.2	9
Web Browsing	15	9	4
Games	9	5.4	5
Other	10	6	2

The utilization percentage column is the specific percentage breakdown gathered from the study done by BI Intelligence [7]. The utilization minutes column describes the actual minutes a typical user would have spent in a day. The derived minutes are based on the utilization percentage in an hour. The actual minutes column shows the exact amount of time spent for each activity for the purpose of this case study. The phone used for the purpose of this experiment did not have a mobile service provider, thus the

talking activity could not be performed. Also, this experimental setup is only simulating one particular use case of a typical smartphone user; by no means does it represent all of the average smartphone users.

This concludes the experiment setup section. The next section discusses interesting findings and results of this data collection process.

4.2 Results

The collected results are analyzed based on three categories: resource utilization, resource access intensity, and activity based sequential analysis. Specifically, the passive data collected are analyzed and compared to the active data. Interesting behaviors and discoveries are discussed in this section.

4.2.1 Privacy Sensitive Resource Utilization

The motivation behind sensitive resource utilization observation is to profile each individual third-party application process and/or the category of application in terms of access volume and accessed resource. In terms of privacy sensitive resource utilization, third-party application process access levels and categories of third-party application process access levels, as well as resource access levels are analyzed. These analyses were run on both passive and active data collected.

Figure 4.1, is a display of resource access attempts made by third-party applications in a passive mode. The x-axis shows the active applications that attempted sensitive resource access. The y-axis marks the number of attempts made in the four-hour period. As it can be observed from the figure, the Weather Channel application followed by Android system processes dominates passive resource activities. In the four-hour span, 234 sensitive resource accesses were made by the Weather Channel application. Then, the total resource access record was followed by Facebook application and Google Play processes. Applications that utilize the least amount of resources are Sleepbot, Pandora, Nike Plus, FIFA, Buzzfeed, and Google Quicksearch.

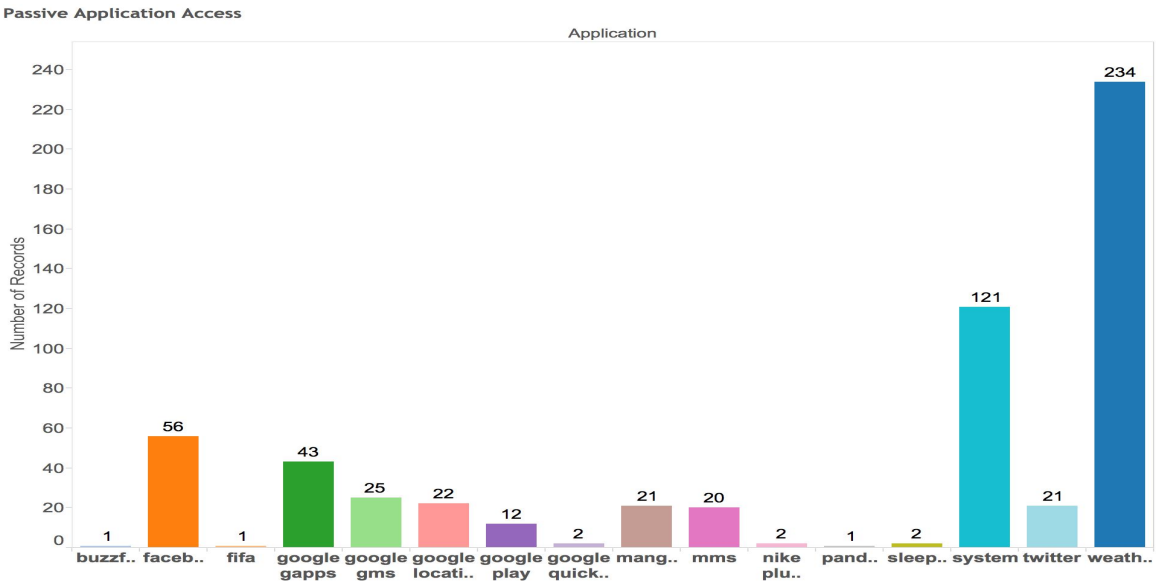

Figure 4.1: Number of sensitive resource accesses made by each third-party application passively.

On the other hand, Figure 4.2 displays the resource access attempts made by third-party applications in a active mode. The figure is displayed in the same format as Figure 4.1. As can be observed in Figure 4.2, the total number of resource accesses is dominated by the Weather Channel application. Applications like Facebook, Twitter, Google services, and Android system processes follow right after. Applications with the least amount of utilization are: Keyboard, Phone, FIFA, Sleepbot, Nike Plus, Pandora, and LINE. One interesting observation in the active case is that, numerous third-party applications have separate process names for different services being carried out. For example, Twitter has two different processes shown in Figure 4.2. One of the process is named com.twitter.android and the other is named com.twitter.android:MediaService.

Overall, comparing the application access graphs of the passive and active case, many observation can be made. In the active case, Figure 4.2, the volume of activity and number of third-party applications are more than that of the passive case. Thus, it could be concluded that third-party applications are partially driven by user interaction. By observing the passive case, Figure 4.1, it could be also concluded that third-party applications tend to have

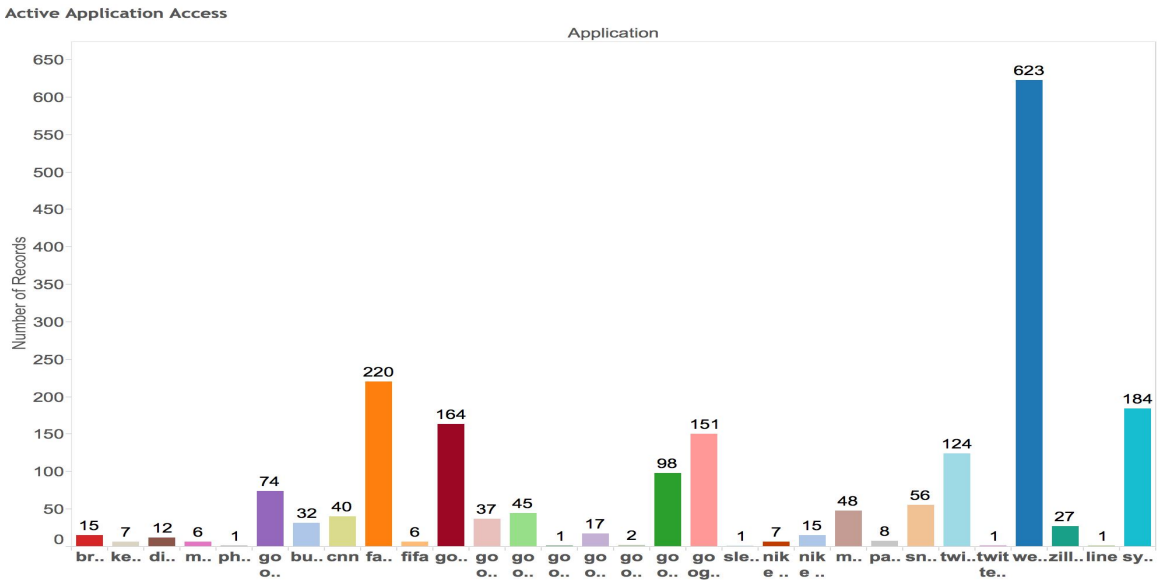

Figure 4.2: Number of sensitive resource accesses made by each third-party application actively.

processes that run in the background even when the application was never initiated in the foreground. In summary, the volume and number of third-party application processes are partly driven by user interaction and partly driven by the applications themselves.

To observe the correlation between the category of applications and the number of resource access attemps, the third-party applications found in Figure 4.1 and 4.2 are classified into different categories. The categorization is mainly based on Table 4.1. The identified categories are Android, entertainment, Google, lifestyle, news, references, social and weather. Some of the categories such as Android and Google are categorized based on the different processes collected since there are no specific applications downloaded for them.

Figure 4.3 shows the number of sensitive resource passive access attempts achieved by each defined category. The x-axis displays the defined categories, and the y-axis displays the number of resource access attempts. It could be observed that the weather category has the most resource access attempts. The Android category follows. Application categories that utilize the least amount of sensitive resources in the passive mode are news,

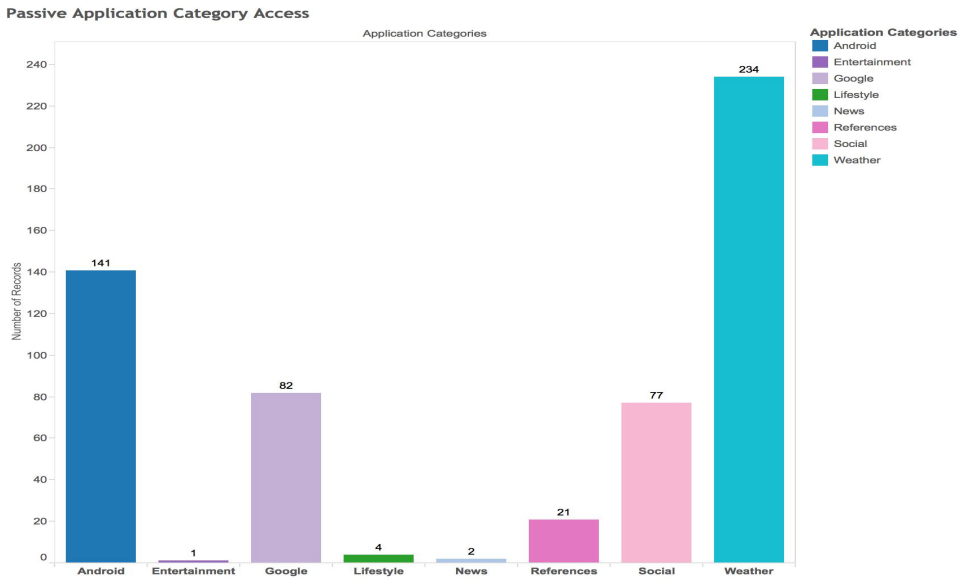

Figure 4.3: Number of sensitive resource accesses made by different categories of third-party application passively.

lifestyle, and entertainment.

For the active case the number of sensitive resource access attempts achieved by each defined category was shown in Figure 4.4. The graph layout is the same structure as Figure 4.3. In Figure 4.4, the weather and Google categories dominates the graph followed by social and Android categories. The minimum sensitive resources accessed in the active mode is the entertainment category.

Comparing both the passive and active case, in Figures 4.3 and 4.4, the difference seem to be the number of resource access attempts. Like resource access attempts by each individual third-party application, the resource access attempts achieved by different categories of applications are also driven by user interaction. The more a certain type of third-party application was interacted, the higher resource access attempt counts will be. By comparing the passive and active case, an understanding of the type of user interaction can be defined. For example, the extreme increase in categories such as Google and social lead to the inference that during that time frame the user has interacted with many social applications as well as Google services related application. This is validated by observing Table 4.2. The type of

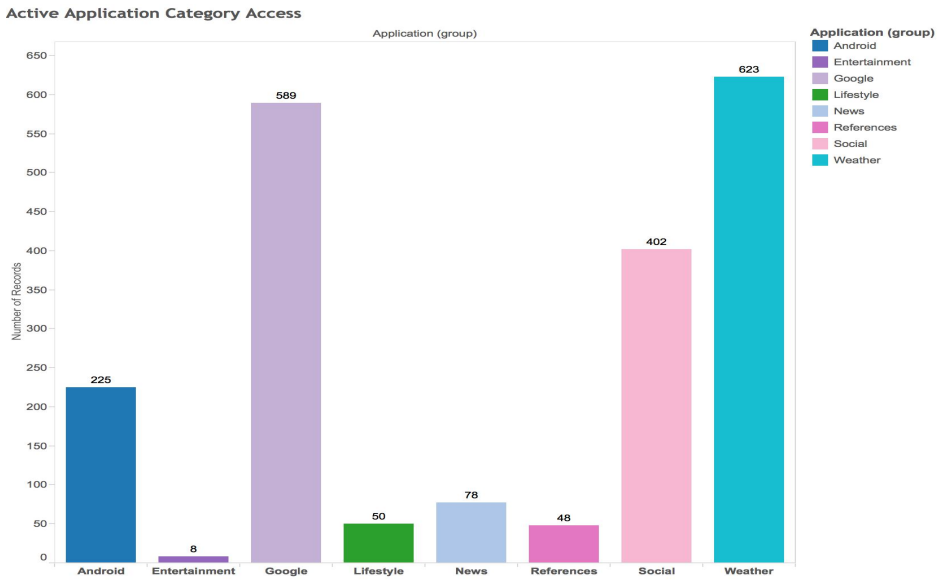

Figure 4.4: Number of sensitive resource accesses made by different categories of third-party application actively.

activity with the most minutes spent is social, thus, the extreme increase in resource access attempts in the social category makes sense.

Figure 4.5, shows the passive access count attempts in terms of resources utilized. The x-axis shows the type of sensitive resources accessed, and y-axis shows the number of access attempts made. The color scheme corresponds to the categories of applications previously defined. Location, address book, and GPS location are resources that are being utilized the most. Most of the location, address book and GPS location resources access attempts are made by application categories such as weather, social and Google. Device identification related resources seem to be accessed a moderate amount as well. Applications in the Google and reference categories seem to utilize device identification related resources the most.

For the active case, the access count attempts in terms of resources utilized are shown in Figure 4.6. Like Figure 4.5, the x-axis shows the type of sensitive resources accessed, and y-axis shows the number of access attempts made. The most utilized resources are location, address book and GPS location. One interesting observation is that third-party applications tend to access location, address book, and GPS location all at once. For that

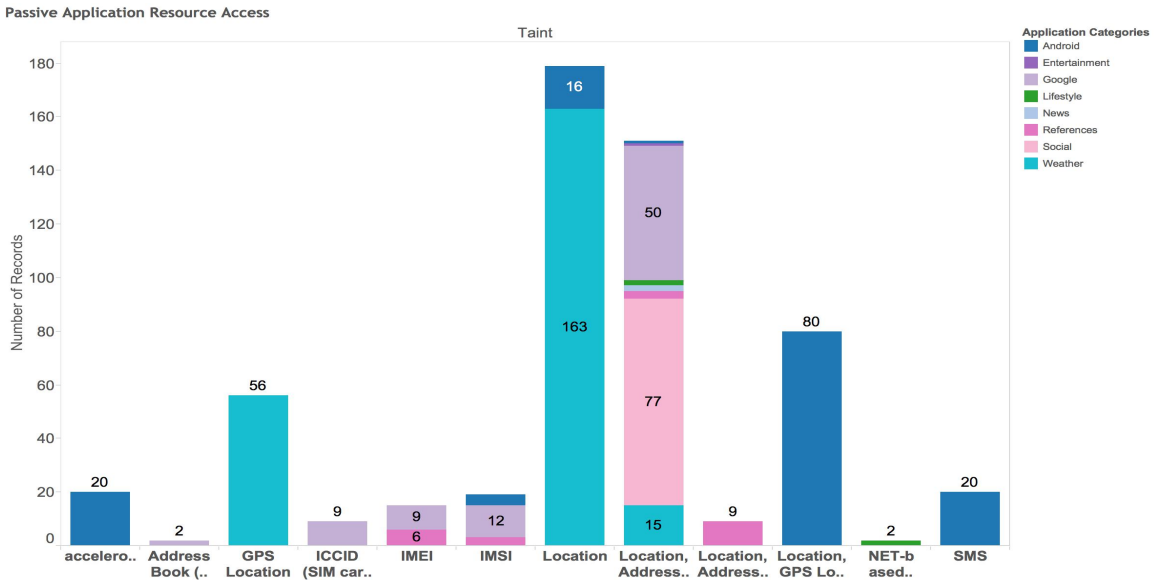

Figure 4.5: Passive utilization of types of resources in terms of categories of third-party application processes.

specific column, all categories of resources utilized it.

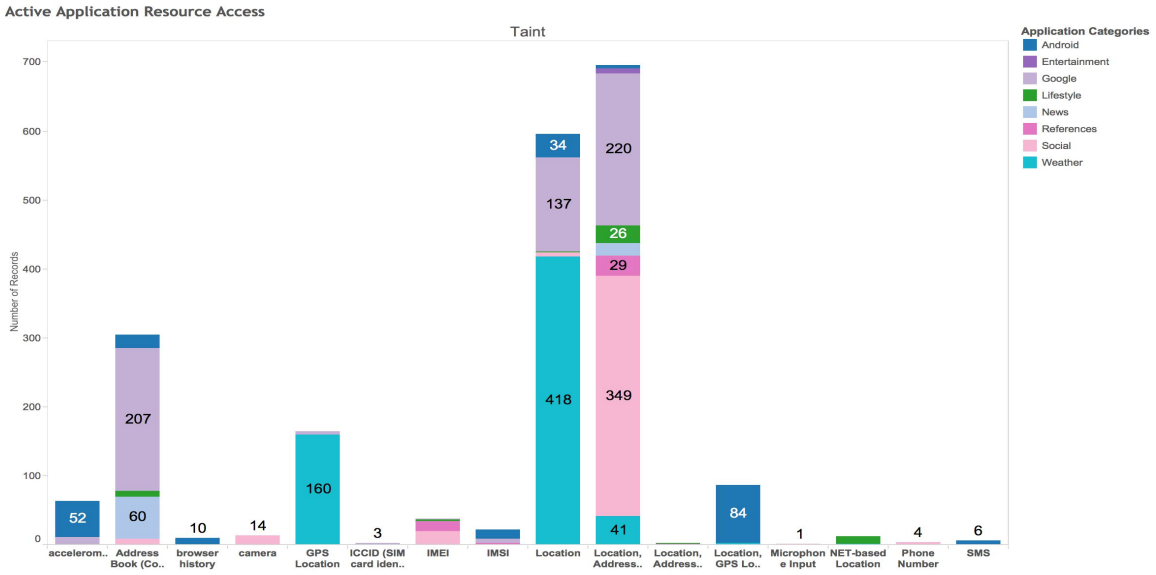

Figure 4.6: Passive utilization of types of resources in terms of categories of third-party application processes.

Comparing the passive and active case of access count attempts in terms

of resources used, application usage drives the type of resources accessed. In the active set, there are a total of four resource groups added. The actively triggered resources are: browser history, camera, microphone input, and phone number. The type of user activity triggered during the experiment drives these extra resources.

Overall, it can be concluded that user interactions drive the type and volume of resource access attempts. This is demonstrated through resources access counts in terms of third-party application processes, categories, and resources. The resources being accessed do not necessarily belong to a specific type of applications; however, there are special cases where a certain type of application only access specific resources. The Weather Channel application is an example. This resource utilization analysis could be used to profile categories of applications. The comparison of passive and active data can also be used to determine user actions. The next subsection discusses intensity in terms of sensitive resource access.

4.2.2 Privacy Sensitive Resource Access Intensity

This section examines the resource access intensity of different categories of applications. The analysis first observes the heat map of resource access intensity in the passive and active case. Then, the analysis is followed by a comparison of total, application, and resource access intensity. Next, the access intensity of each application category is then graphed to observe patterns and behaviors. Finally, specific cases of the impact of user-interaction to access intensity are observed.

Figure 4.7 displays the passive access intensity of applications and their utilized resources in a heat map format. The left column displays the categories of applications. The first row shows the type of sensitive resources accessed. The number of access attempts made in a four-hour period determines the intensity. The weather category is the most access intensive, then followed by Android, Google, and social applications. Application categories such as Android, Google, and references tend to access multiple resources, however, with minimum intensity for each resources accessed. The most intensively accessed resource is location-based resource.

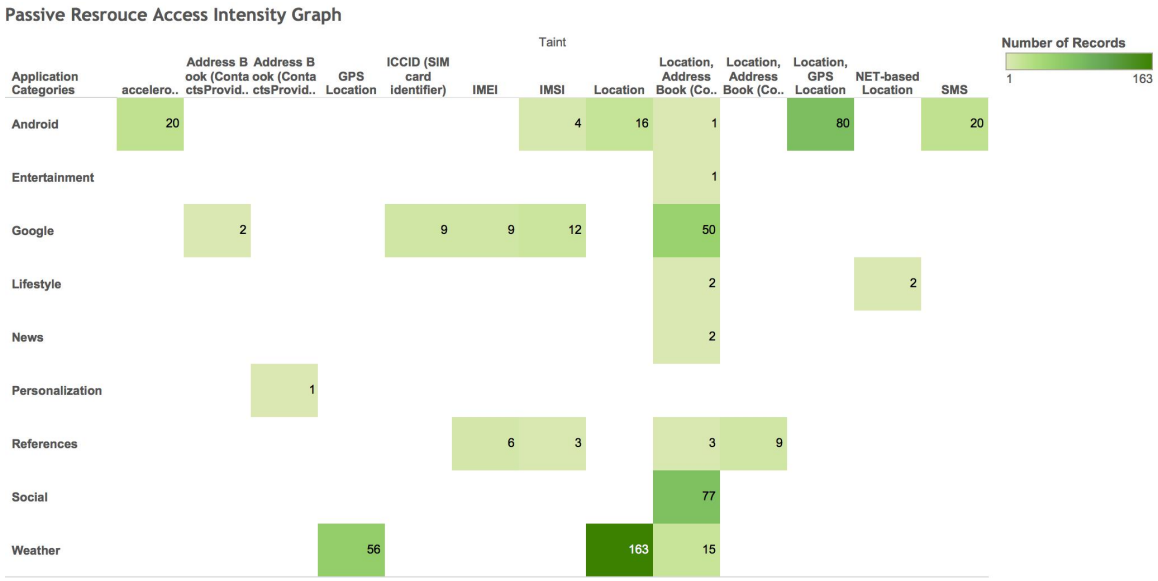

Figure 4.7: Intensity heat map of passive resource access attempts.

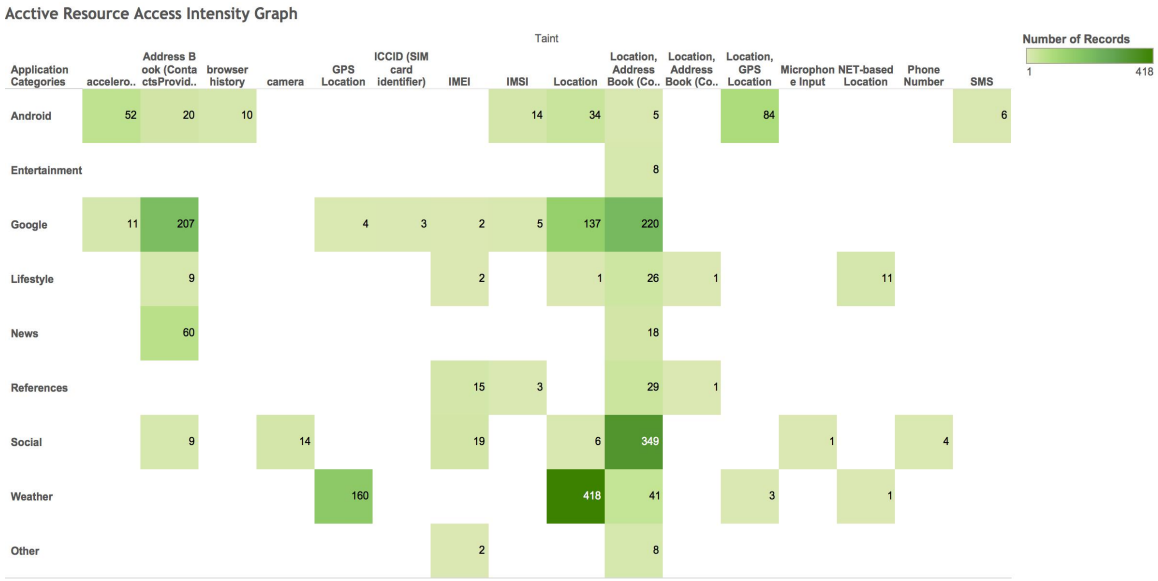

Figure 4.8: Intensity heat map of active resource access attempts.

Like the passive case, Figure 4.8 shows the active access intensity of applications and their utilized resources. In the active case, most of the activity seems to hover around location and address book access. The categories with high intensity access are Google, social and weather. The rest

of the access intensities are pretty spread due to the type of activity that was performed during the experiment.

Comparing the passive and active case, there are many differences. With each triggered activity, there is an increase accessed resource type. For example, social category in the passive case accessed only location, address book, and GPS. In the active case, resources such as camera, IMEI, microphone, and phone number are all resources accessed in addition. The heat map of the active case has a more spread-out look and dominated by multiple high intensity areas. In summary, user induced actions drive the type of resources accessed and the access intensity.

Figure 4.9 displays the passive intensity comparison of total access, application category access, and resource access over times. The y-axis shows the number of access attempts. The x-axis is the time line of the experiment. The top, middle, and bottom graphs display the total application access intensity; application categories access intensity, and resource access intensity overtime respectively. For all three graphs, there is an initial sort of access, and this is due to the initial startup of the Android system. Overtime, that extreme intensity dies down and peaks again periodically. The periodic peaks are mainly driven by the weather application acquiring location-based access.

For the active case, the intensity comparison is displayed in Figure 4.10. Here, sharp peaks that occur randomly characterize the behaviors of different access intensities overtime. The randomness is mainly driven by user interaction at that specific moment. Categories with extreme high peaks are Google and social. The types of resources that seem to be accessed during those times are address book, location, and GPS. In summary, the exact moment of user interaction determines the access intensity of application category as well as resource accessed.

Figure 4.11 is an expanded view of the access intensity of each application category. The y-axis shows the categories, and the x-axis displays the time. Categories such as entertainment, lifestyle, and news behave in the

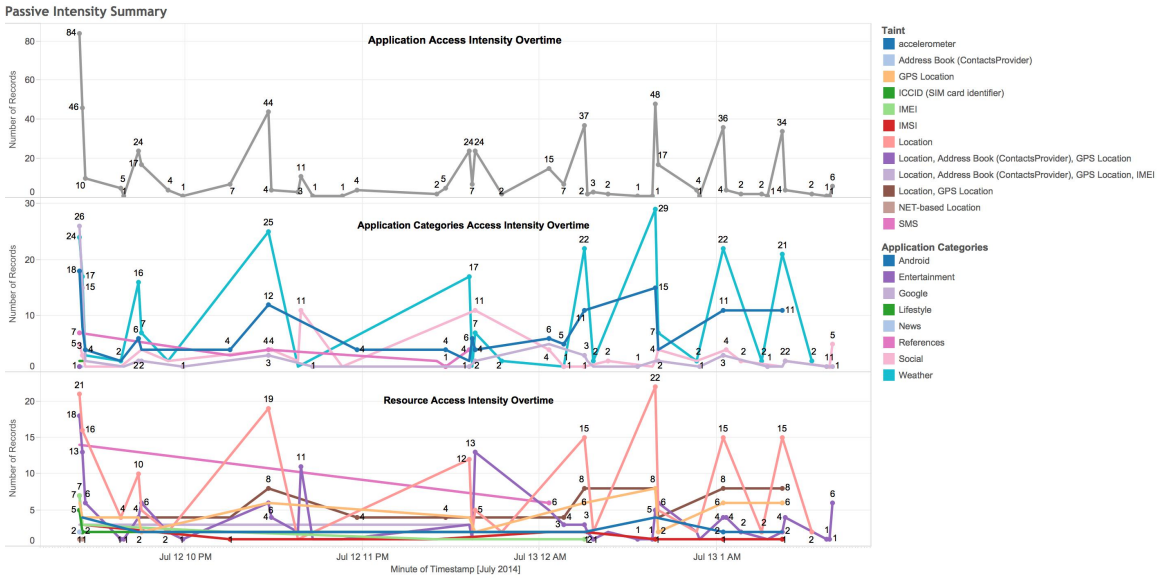

Figure 4.9: Intensity summary graph of passive resource access attempts.

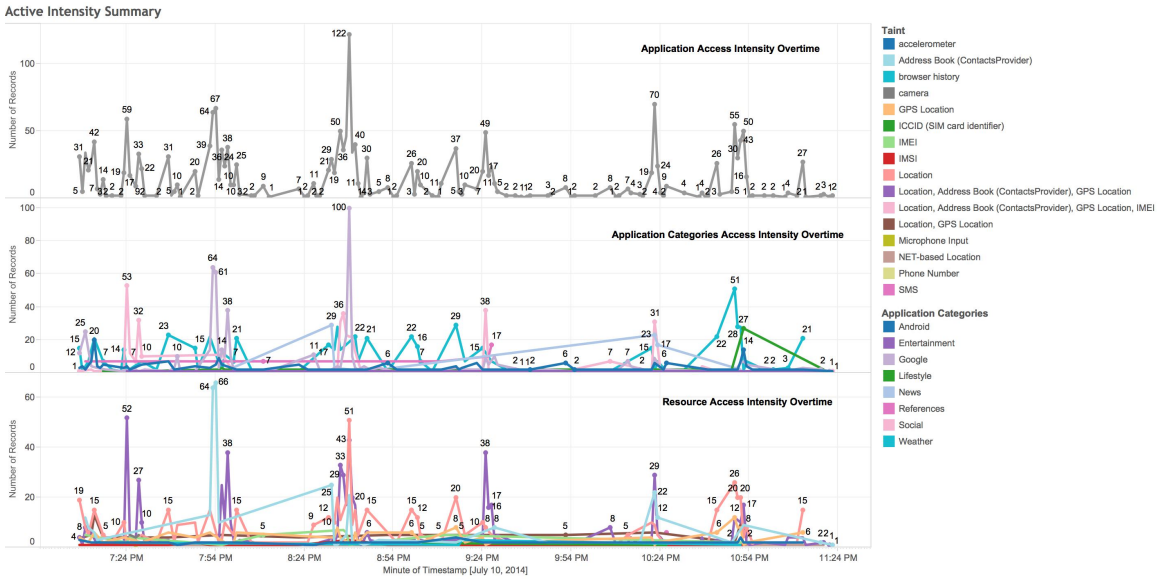

Figure 4.10: Intensity summary graph of active resource access attempts.

same way. These categories have single point access intensity. Google category behaves with an initial drop. The access intensity of Google drops dramatically from initialization then flat lines with minor activities. More continuous access intensity is exhibited by categories such as social, weather,

and Android. Periodic peaks characterize them. The reference category seems to have an obscure behavior. This would require further analysis.

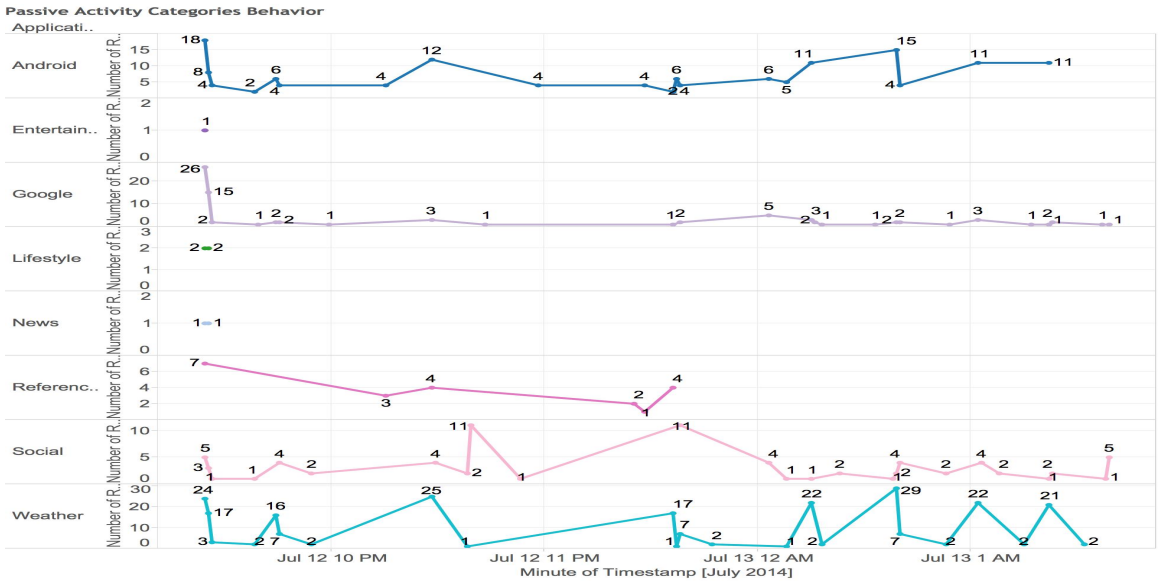

Figure 4.11: Intensity of passive resource access attempts by application categories.

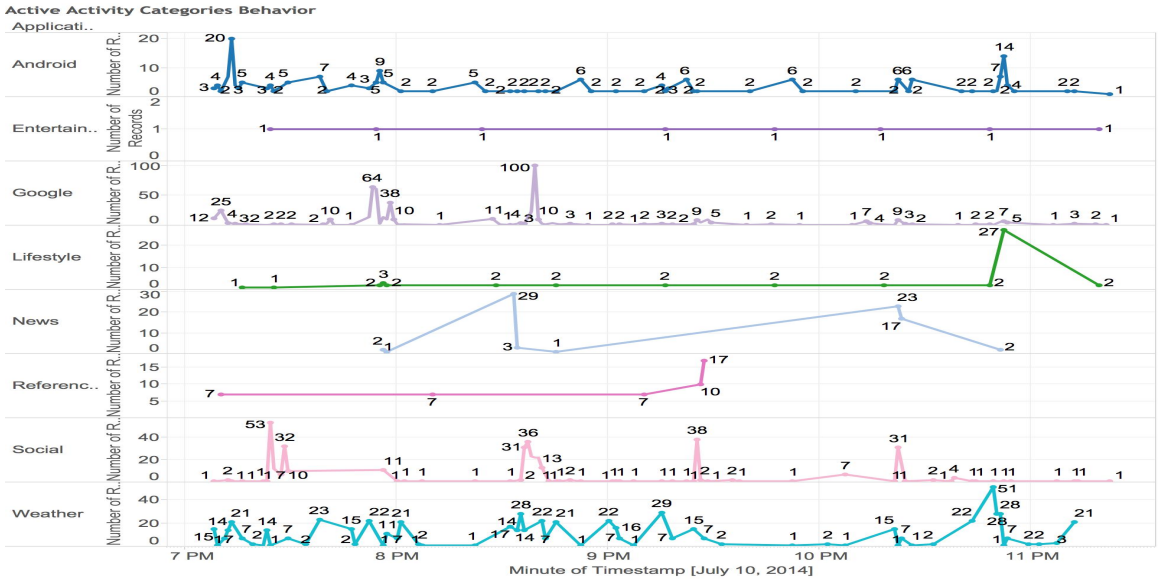

Figure 4.12: Intensity of active resource access attempts by application categories.

The active case of the access intensity by application categories is displayed in Figure 4.12. In the active case, categories such as entertainment, lifestyle, news, and reference behave the same way. For them, there are periodic sensitive resource access attempts when the application is not running in the foreground. These periodic sensitive resource access attempts are then followed by extreme peaks recording an event where the user has actively interacted with these applications. Application categories such as Android, Google, and social behave in another way. These categories are characterized by small continuous resource access attempts followed by extreme peaks signaling user interaction. Lastly, the weather category graph resembles a zigzag line. It is characterized by close high resource access intensity followed by low resource access intensity. Overall, the active case is driven by user interaction, and it builds on the intensity of the passive case.

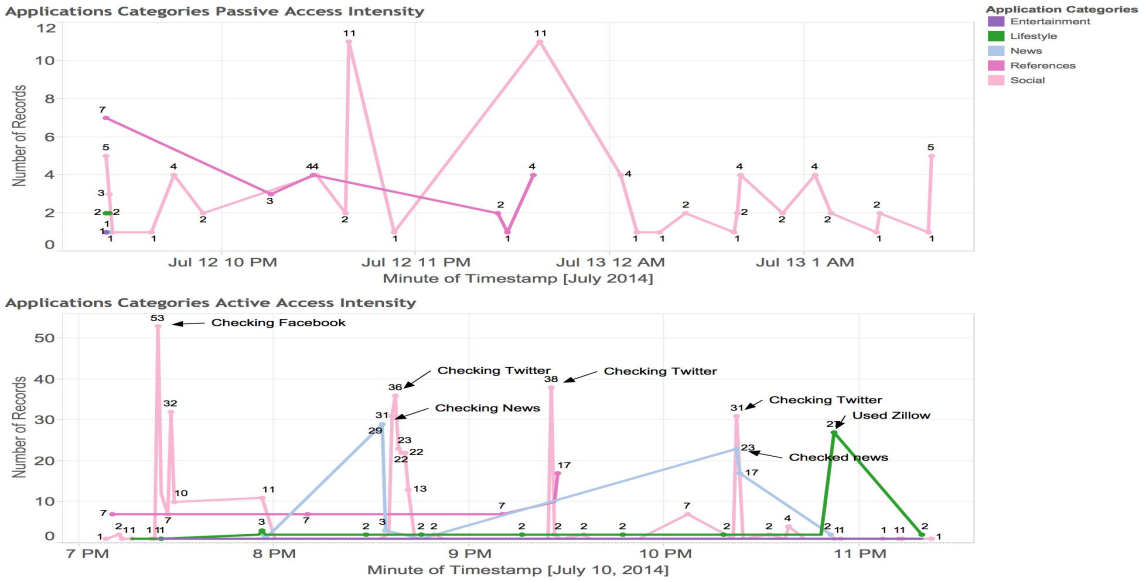

Figure 4.13: Intensity comparison between specific passive and active application categories.

Figure 4.13 details specific passive and active application categories. The y-axis is the number of access attempts, and the x-axis is the time. The chosen application categories are entertainment, lifestyle, news, reference, and social. The top graph displays their behavior in the passive case, and the bottom graph displays their behavior in the active case. Each activity

performed, they were also marked on the active graph. As seen on the graph, user interactions have a correlation with sensitive resource access intensity.

Throughout intensity analysis, there are a few things that can be observed. First, each application category has its access intensity behavior. This access intensity behavior differs between the passive and active case. They differ in the sense that, the active access intensity builds on existing passive access intensity model. Finally, much like resource utilization, the access intensity level is also driven by user interaction.

4.2.3 Network Traffic Analysis

Aside from sensitive resource utilization and access intensity analysis, the data collected also enable network traffic analysis in terms of application, application categories, and resources. This section discusses the outbound network traffic that contains sensitive information.

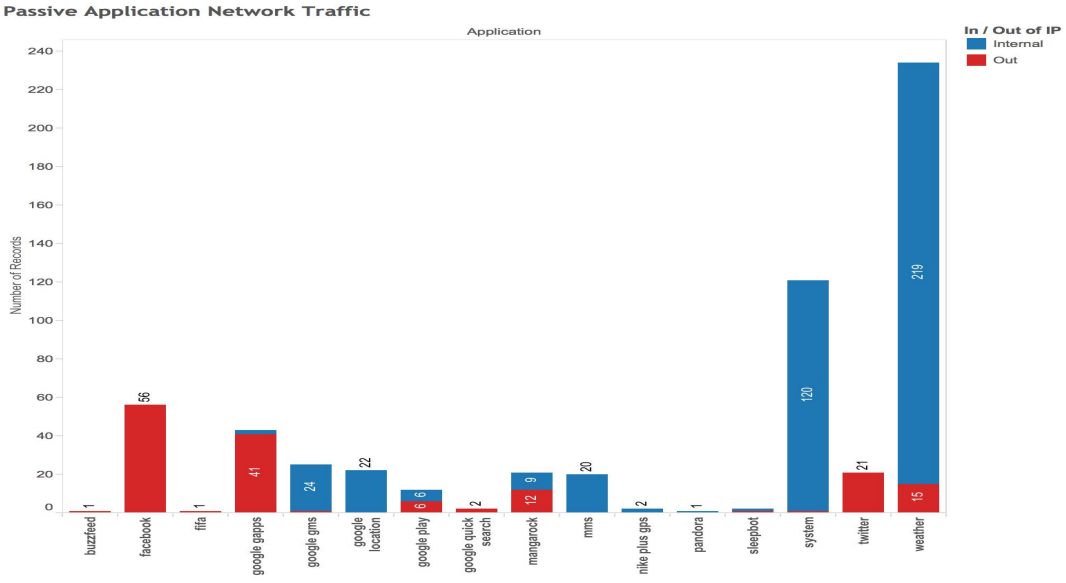

Figure 4.14: Passive application network traffic.

Figure 4.14 displays the passive application network traffic. The y-axis shows the number of resource access attempts, and the x-axis displays the names of applications. The color scheme displays application network traffic. Red color represents the out flow of network traffic, and blue represents

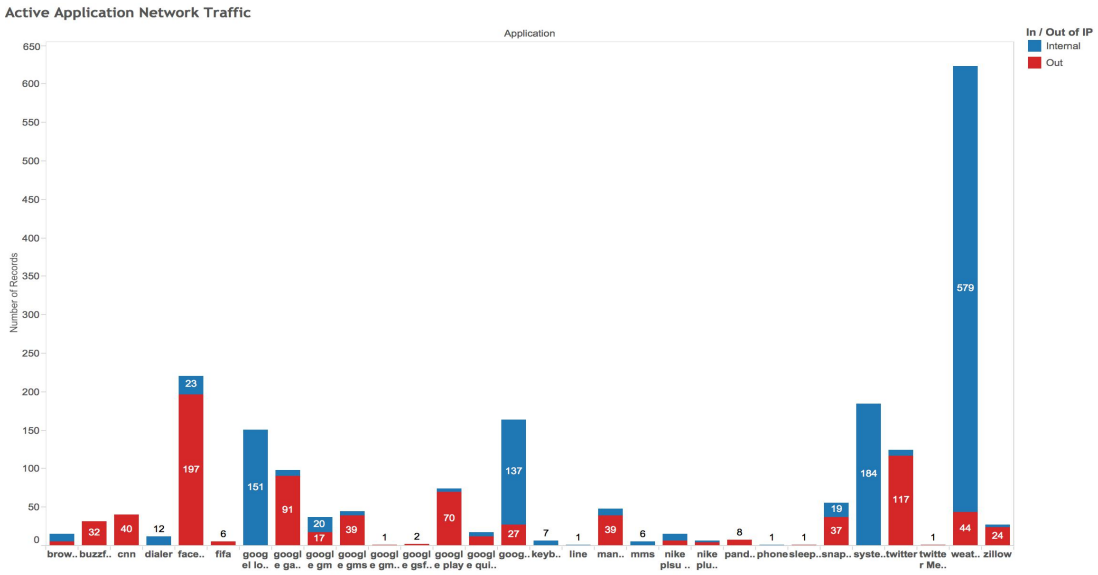

Figure 4.15: Active application network traffic.

internal resource access. As could be observed, most resources have both internal and external activity. However, Facebook and Twitter seem to have only outbound traffic. Facebook is the most outbound intensive application.

The active case is displayed in Figure 4.15. With user interaction, there is an increase in outbound network traffic. For the most part applications have both internal and external traffic. Applications such as Facebook and Twitter that did not have an internal traffic shown in Figure 4.14, now have both internal and external traffic shown.

In terms of application category, the graphs summarize the network traffic of each individual application. The passive category analysis is displayed in Figure 4.16. The most outbound traffic intensive application category is social, where one hundred percent of the resource activities were outbound traffic. On the other hand, weather and Android application categories are mostly dominated by internal activities.

For the active case, the results are displayed in Figure 4.17. From the figure, it is observed that there is an increase in outbound traffic as well as internal activity for each application category. Google category seems to have a balanced internal and external traffic. All other user triggered activity categories seem to have an extreme increase in outbound traffic. For

example, the news category changed from two in the passive case to seventy-eight in the active case. Definitely, the type of user action is positively related to outbound network traffic activity.

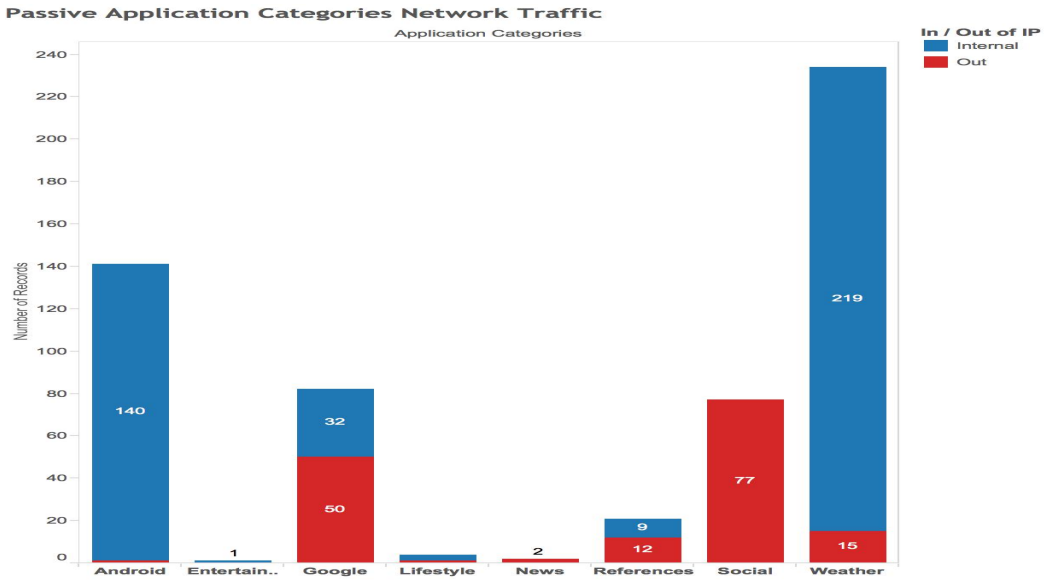

Figure 4.16: Passive application network traffic in terms of application categories

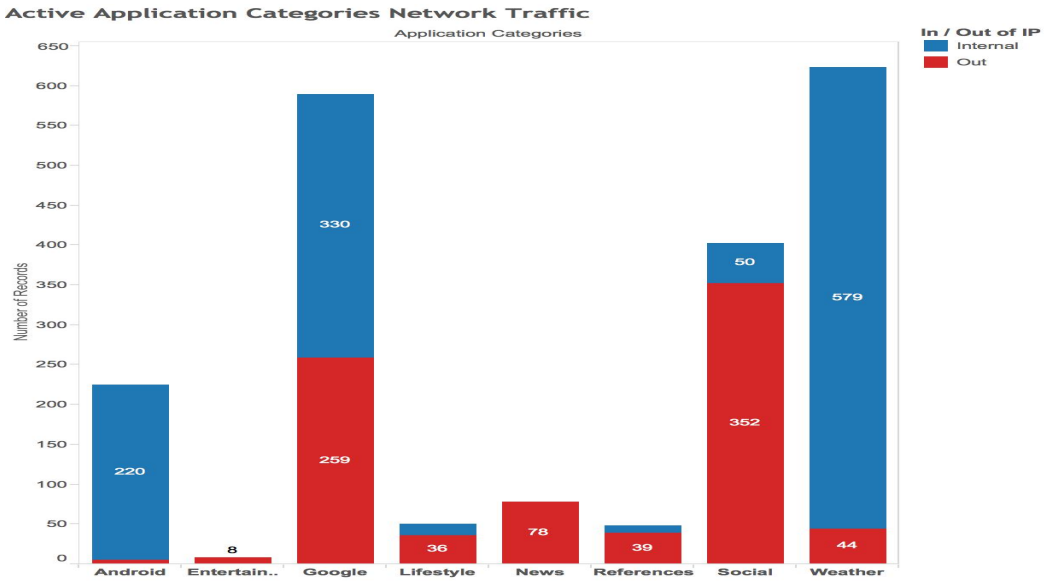

Figure 4.17: Active application network traffic in terms of application categories.

Aside from network traffic analysis based on application categories, Figure 4.18 shows the passive network traffic behavior in terms of utilized resources. The y-axis shows the number of sensitive resource access attempts, and the x-axis shows the type of utilized resource. It is very clear from the graph that location, address book, GPS resource combination has the most outbound traffic. location, address book, GPS, and IMEI combination has the second highest outbound traffic. In the passive mode, most utilized resources are for internal purposes, and kept internal.

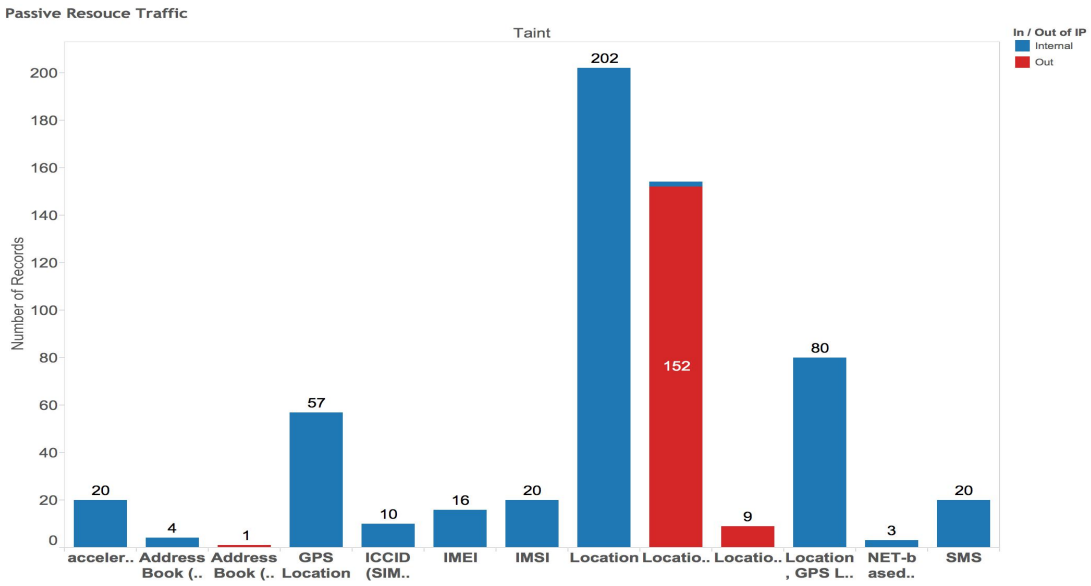

Figure 4.18: Passive application network traffic in terms of utilized resources.

Figure 4.23 shows the active application network traffic in terms of utilized resources. Very much like the passive case, resource combination such as location, address book, and GPS dominate outbound traffic. However, with specific user activity, there is an increase in outbound traffic for address book. Also, resources such as IMEI that did not have outbound traffic is now shown with activities. Overall, the outbound traffic intensity and type of sensitive resource depends on the type of action performed by the user.

Besides the traffic volume, Table 4.3 displays the unique findings in terms of IP addresses. It is observed, for each application, there are multiple IP addresses assigned. For example, mangarock has a total of eleven IP addresses assigned for outbound traffic in the passive case. However,

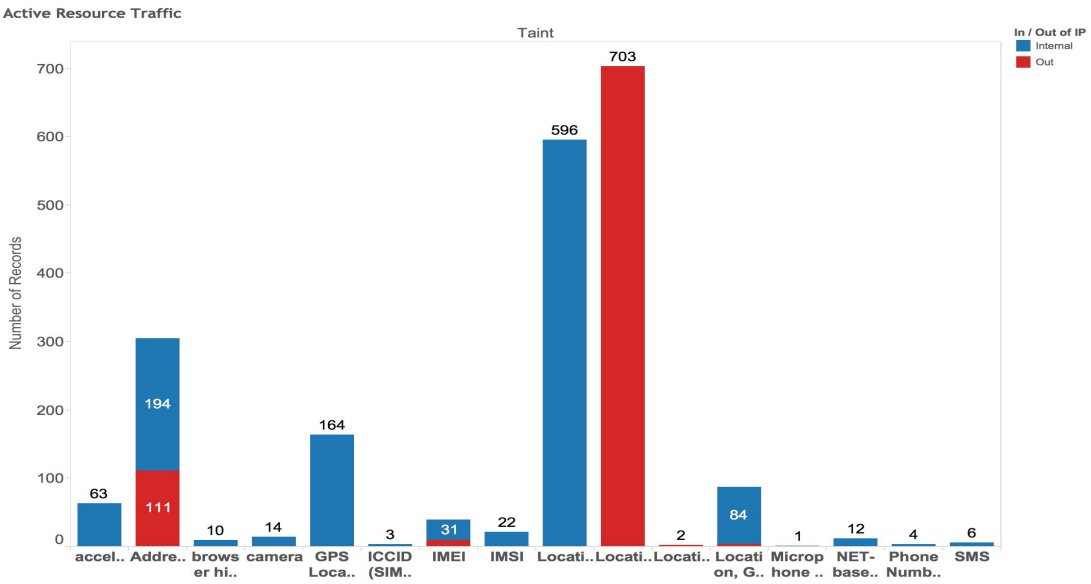

Figure 4.19: Active application network traffic in terms of utilized resources.

with an increase in user activity, there is an increase in number of unique IP addresses. As shown in the passive and active IP address count columns, the number of unique IP addresses increased. Facebook application increased from five IP addresses in the passive case to twelve IP addresses in the active case. Some applications such as Zillow went from no external network activity to nine unique IP addresses in the active case.

After looking at the unique IP addresses and the applications sending to those IP addresses, it was quickly observed that there are certain IP addresses that are used by different applications. The interesting findings are shown in Table 4.4. The IP address column displays the unique IP addresses. The purpose column displays the owner of the IP address and the location of it. The application column displays the applications that are utilizing that specific address. For example, Buzzfeed and Facebook utilized the same IP address. Since the used IP address is assigned to Facebook, it seems like the application Buzzfeed is also sending information to Facebook. This might suggest collaboration between Facebook and Buzzfeed.

Overall, there are a few things that could be summarized about network traffic. First, outbound network traffic usually carry location, address book, and IMEI information. The volume and type of outbound network traffic

Table 4.3: Number of IP address for each application in the passive and active case.

Application	Passive IP Addr. Count	Active IP Addr. Count
browser	-	4
buzzfeed	1	9
cnn	-	9
facebook	5	12
fifa	1	4
google gapps	6	7
google gm	-	2
google gms	1	6
google gms maps	-	1
google gsf login	-	2
google play	1	19
google quick search	2	4
googlevoice	-	2
mangarock	11	15
nike plus gps	-	3
nike plus gps remote	-	6
pandora	1	8
sleepbot	2	1
snapchat	-	2
system	1	-
twitter	7	18
twitter MediaService	-	1
weather	11	29
zillow	-	9

Table 4.4: Applications with same IP addresses.

IP Address	Purpose	Application
IP 1	Akamai Technologies (Cambridge, Massachusetts)	FIFA, Nike Plus GPS
IP 2	Google Inc. (Mountain View, California)	Google gms, Google Voice
IP 3	Facebook (Palo Alto, California)	Buzzfeed, Facebook
IP 4	Google Inc. (Mountain View, California)	Google gms maps, Zillow
IP 5	Amazon AWS (Seattle Washington)	Buzzfeed, Mangarock
IP 6	Amazon AWS (Ashburn, Virginia)	Pandora, Sleepbot

is usually driven by user interaction. Each installed third-party application utilizes multiple unique IP addresses, and with increase in user activity, the

number of unique IP addresses increase as well. Finally, there are also special cases, where the same IP addresses are used by different applications. This behavior might suggest collaboration between third-party applications outside the mobile device. The next section discusses action based sequential analysis.

4.2.4 Action Based Sequential Analysis

The data collected also provide insights into typical behaviors of specific actions taken. The collected data were first sectioned off by the application name. Then, depending of the timestamps for each data entry the resource type of recorded data were analyzed to observe repeating pattern. Specifically, the Google location process, Snapchat process, and the Weather Channel processes were carefully observed and analyzed.

For analyzing the Google location process, it is observed that the process behaves differently from the passive and active modes. Figure 4.20 displays the passive mode sequence of Google location process. Initially, the accelerometer sensor is added. After, the process checks for passive network provider as well as mobile network provider. Then, location update requests are made with a looper. The looper handles massage loop responses. This cycle continues every now and then throughout the passive mode.

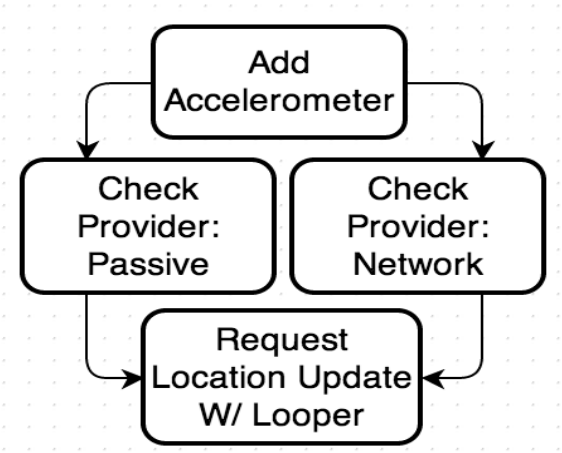

Figure 4.20: Passive action sequence of Google location process.

For the active Google location process sequence, the behavior can be seen in Figure 4.21. At first glance, the active cycle sequence looks like it adapts the already existing passive sequence. First, accelerometer is added, then network providers are checked, and location update request is made with a callback looper. The additional cycle actions are checking for GPS provider and finally registering an accelerometer with a callback handler. This is a typical cycle sequence of actions for Google location process in the active mode.

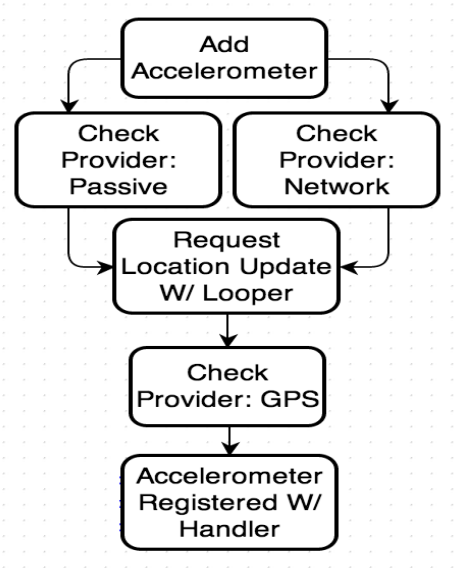

Figure 4.21: Active action sequence of Google location process.

Through the observation of both the passive and active behavior of Google location process, there is one interesting observation. In normal cases, processes active behaviors tend to build on the behaviors of the processes in the passive case.

Another application action analyzed is the Snapchat process. The sequence observed is displayed in Figure 4.22. The specific user action taken was first start Snapchat, take a picture, and send off the picture taken to a friend. The actual results collected show the sequence of sensitive resources accessed. First, the device IMEI number is obtained, and then depending on the user, the front or back camera is accessed. After, the phone number of the device is also collected. When a picture is taken, the camera preview

resource is used, and the camcorder is registered. Finally, the picture is sent out using an external IP address.

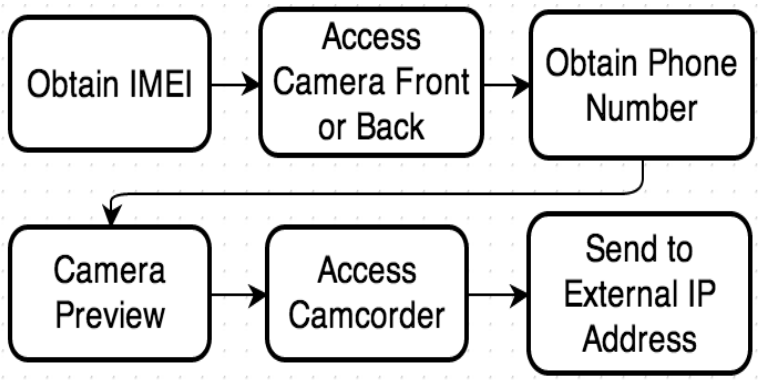

Figure 4.22: Action sequence of using Snapchat.

Aside from the analyzed sequence, the outbound data were analyzed as well. In the Snapchat case, the outbound data consisted of information such as user ID, device information, photo ID, screenshot count, and timestamp. Most of the information seems to be critical for application development and business analysis.

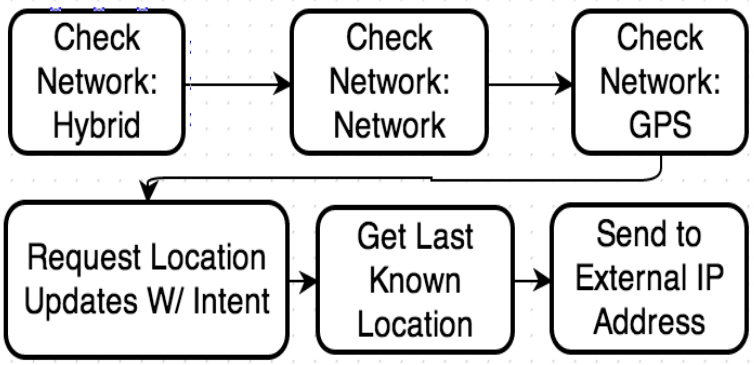

Figure 4.23: Action sequence of the Weather Channel application.

The last analyzed application is the Weather Channel. The resulted sequence is shown in Figure 4.23. For the sequence, the Weather Channel application utilized the location resource first by checking available network accesses. In this case, the application checks for hybrid network provider

followed by mobile network provider and GPS provider. Once a provider is verified, the process requests for location update with a callback intent registered. After, the process gets last known location and sends the acquired location to an external IP address. This sequence process is very consistent throughout both the passive and active modes. The location information sent out is usually just the coordinates obtained from the get last known location step.

Overall, the specific action cases discussed in this section demonstrate the capability of sequential action profiling for user actions from the data collected. There are three main observation made from the analysis. First, active resource access sequences tend to build on that of the passive sequence. Secondly, for location based resource accesses, network provider checking seems to be a priority. The sequence of network provider checking is as follows: hybrid, network, and GPS. Finally, through observation of the actual outbound data, it is shown that most outbound data contain application specific information that is crucial for application development and business analysis.

4.3 Performance

The original designe of the system was for the purpose of collecting communication information between third-party applications and available resources. This is an initial work in this area, thus, not much performance emphasis was given for the overall system. The performance analysis that was evaluated was based on data usage and power consumption. Table 4.5, shows the data and battery usage evaluation.

Measures	Time Frame	Stock ROM	Panorama	Expected Usage Per Recording
Data Usage	24 hours	13MB	21MB	0.006MB
Battery Power	6 hours	11 percent	18 percent	0.021 percent

Table 4.5: Performance measures of Panorama.

In terms of data usage, a user can expect an 8MB increase every day. For power consuption, a user should expect a 7 percent increase for every six hours. This presents an opportunity for future improvement in this area.

Chapter 5

Conclusion and Future Work

Data is, now more than ever, becoming a driving force behind technology developments as well as business progression. With the increase in market growth for smartphone devices, and the increase human dependency on their smartphone devices, mobile data is becoming a crucial piece of information for various aspects of analysis. There are cases, where mobile data has been used to understand human communication patterns [21], social relationships [8], and etc. However, there are few researches that focused on analyzing the behavior of applications that are installed on the device.

Given the need and benefits of background services accessing mobile resources, users often have little understanding of the leakage of security and privacy information. To address these risks, datasets should be collected relating to activities of background services accessing privacy sensitive mobile resources. There are also many challenges involved with this type of data collection. The first challenge is the Android security structure. The sandbox structure prevents the monitoring of other third-party applications. The second challenge is identifying and actually recording privacy sensitive mobile resources access events. The third challenge is that the collection process must seamlessly work in parallel with the Android system.

With the challenges defined, the resulted system designed is a three component privacy sensitive resource access monitoring for Android system. The sandbox security structure challenge was mitigated by designing a collection mechanism that modified the framework layer of the Android software stack and incorporated the TaintDroid sensitive information flow tracking system. The recording of resource access challenge was overcome by implementing resource specific action listeners through the utilization of

the TaintDroid logging system. Finally, the seamless architecture challenge was overcome by embedding the listeners in each of the sensitive resource components. By embedding the listeners in sensitive resource components, third-party applications do not require any additional implementation for sensitive resource accesses to be captured. Here are some of the highlights of the system:

- Enabled dynamic monitoring and collection of sensitive resource activity
- Provided modular framework for easy modification
- Provided understanding of third-party application behavior in passive and active case
- Enabled sequential analysis of specific third-party application

Overall, the system enabled dynamic monitoring and collection of sensitive resource activities. It also provided a modular framework for easy modification in the future.

The data collected as a result of the overall system provided insights into the behaviors of third-party application. The privacy sensitive resource utilization analysis showed that third-party applications do utilize sensitive resources passively. It also showed that location based resource is the most utilized sensitive resource. In the intensity analysis, it is discovered that each activity type has its access intensity behavior. Also, each active sensitive resource access intensity builds on its existing passive sensitive resource access intensity model. In the network traffic case, outbound traffic were usually composed of location, address book, and IMEI based information. Another interesting finding is that there are many third-party applications that actually utilize the same IP address. Lastly, in the action based sequential analysis, it was discovered that certain accessed resource follow a typical access sequence. For example, for location-based resource, the access sequence is usually checking for hybrid provider, then network provider, and finally GPS provider.

The analyzed results provided insights into third-party applications and sensitive resources that were often overlooked. However, there are many

limitations with this data collection tool. Due to the placement of some of the sensitive resource access listeners, it is very difficult to determine the application that initiated the action. For example, the camera resource does not register the initiated third-party application. Another limitation is on the number of sensitive resource monitored. This overall system was designed to demonstrate the capability of the collection mechanism; it does not monitor all the sensitive resources on the mobile device.

As for future work there are many things that could be done in three specific areas. For the collection module, more listeners could be implemented to increase the data sample size. During the collection process, relating the current process and initiated process can also help monitor application-to-application communication. For the server module, a graphical web interface could be created to display the data collected. Further, for data mining efficiency, multiple data tables could be created to represent each monitored application. Lastly, for analysis, the experiment could be tested on multiple devices to observe cross device behavior. Also, this tool could be used to observe application sequential behavior and compare that to static application sequential behavior analysis. Finally, machine learning could be used on the data collected to train devices to identify what is normal and what is abnormal process/resource behavior. Overall, this is a small step in contributing datasets and presenting creative ways that the datasets could be used to solve real world problems.

Bibliography

[1] Android security overview. https://source.android.com/devices/tech/security/.

[2] Android logging system. http://elinux.org/Android_Logging_System, Feb 2012.

[3] Activities. http://developer.android.com/guide/components/activities.html, April 2014.

[4] Application fundamentals. http://developer.android.com/guide/components/fundamentals.html, April 2014.

[5] Most popular google play categories. http://www.appbrain.com/stats/android-market-app-categories, April 2014.

[6] L. Meng C. Poellabauer D. Hachen O. Lizardo A. Striegel, S. Liu. Lessons learned from the netsense smartphone study. *ACM SIGCOMM Computer Communication Review*, 43(4):51–56, Oct 2013.

[7] M. Ballve. How much time do we really spend on our smartphones every day. *Business Insider*, Jun 2013.

[8] W. Dong, B. Lepri, and A. Pentland. Modeling the co-evolution of behaviors and social relationships using mobile phone data. *MUM '11 Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, pages 134–143, Dec 2011.

[9] T. Dyhouse. Growing security threat from mobile phone apps. http://www.computerweekly.com/opinion/The-growing-security-threat-from-mobile-phone-apps, September 2010.

[10] N. Eagle and A. Pentland. Reality mining: sensing complete social systems. *Personal and Ubiquitous Computing*, 10:255–268, May 2006.

[11] D. Weitzner F. Zhang, F. Shih. No surprises: measuring intrusiveness of smartphone applications by detecting objective context deviations. *WPES'13 Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 291–296, 2013.

[12] X. Wei L. Gomez I. Neamtiu M. Faloutsos. Profiledroid: Multi-layer profiling of android applications. *Mobicom'12 Proceedings of the 18th annual international conference on Mobile computing and networking*, (137-148), aug 2012.

[13] J. Fingas. Android tops 81 percent of smartphone market share in q3, Dec 2013.

[14] J. Yang E. Tapia D. Crandall H. Zhang, Z. Yan. mfingerprint: Privacy-preserving user modeling with multimodal mobile device footprints. *Social Computing, Behavioral-Cultural Modeling and Prdiction*, pages 195–203, April 2014.

[15] D. Halper. Nbc: All visitors to sochi olympics immediately hacked.

[16] X. Xie J. Yuan, Y. Zheng. Discovering regions of different functions in a city using human mobility and pois. *KDD'12 Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 186–194, 2012.

[17] E. Jeman. 1 billion smartphones shipped in 2013. *InformationWeek*, Jan 2014.

[18] K. Ali K. Shi. Getjar mobile application recommendations with very sparse datasets. *KDD'12 Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 204–212, 2012.

[19] Aaron Koblin. Amsterdam sms messages on queen's day. 2009.

[20] J. Kwapisz, G. Weiss, and S. Moore. Activity Recognition using Cell Phone Accelerometers. *ACM SIGKDD Explorations Newsletter*, 12(2):74–82, Dec 2010.

[21] V. Kyllonen, J. Mantyjarvi, J. Huhtala, A. Sarjanoja, and J. Hakkila. Mobile Application for Developing Self-Awareness of Personal Communication Patterns. `http://personalinformatics.org/docs/chi2010/hakkila_personal_communication.pdf`, Apr 2010.

[22] W. Peng L. Wei, Y. Zheng. Constructing popular routes from uncertain trajectories. *KDD'12 Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 195–203, 2012.

[23] J. K. Laurila, Daniel Gatica-Perez, I. Aad, Blom J., Olivier Bornet, Trinh-Minh-Tri Do, O. Dousse, J. Eberle, and M. Miettinen. The mobile data challenge: Big data for mobile computing research. In *Pervasive Computing*, 2012.

[24] I. Lunden. Gartner: 102b app store downloads globally in 2013, 26b in sales, 17 percent from in-app purchases, Sep 2013.

[25] A. Mylonas, S. Dritsas, B. Tsoumas, and D. Gritzalis. Smartphone security evaluation - the malware attack case. *International Conference on Security and Cryptography (SECRYPT-2011)*, pages 25–36, July 2011.

[26] Ahas R., Slim S., Jarv O., Saluveer E., and Tiru M. Using mobile positioning data to model locations meaningful users of mobile phones. *Journal of Urban Technology*, 17(1):3 – 27, April 2010.

[27] B. Chun L. Cox J. Jung P. McDaniel A. Sheth W. Enck, P. Gilbert. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. *OSDI'10 Proceedings of the 9th USENIX conference on Operating systems design and implementation*, (1-6), 2010.

[28] C. Lin H. Li X. Zhou X. Wang. Screenmilker: How to milk your android screen for secrets. *21th Annual Network and Distributed System Secuirty Symposium (NDSS'14)*, Feb 2013.

[29] J. Yap. Android to take 58 percent of smartphone apps, May 2013.